Do As I Can, Not As I Say: Grounding Language in Robotic Affordances

Anonymous Author(s)
Affiliation
Address
email

Abstract: Large language models can encode a wealth of semantic knowledge about the world. Such knowledge could be extremely useful to robots aiming to act upon high-level, temporally extended instructions expressed in natural language. However, a significant weakness of language models is that they lack real-world experience, which makes it difficult to leverage them for decision making within a given embodiment. For example, asking a language model to describe how to clean a spill might result in a reasonable narrative, but it may not be applicable to a particular agent, such as a robot, that needs to perform this task in a particular environment. We propose to provide real-world grounding by means of pretrained skills, which are used to constrain the model to propose natural language actions that are both feasible and contextually appropriate. The robot can act as the language model's "hands and eyes," while the language model supplies high-level semantic knowledge about the task. We show how low-level skills can be combined with large language models so that the language model provides high-level knowledge about the procedures for performing complex and temporally extended instructions, while value functions associated with these skills provide the grounding necessary to connect this knowledge to a particular physical environment. We evaluate our method on a number of real-world robotic tasks, where we show the need for real-world grounding and that this approach is capable of completing long-horizon, abstract, natural language instructions on a mobile manipulator. [1]

Figure 1: LLMs have not interacted with their environment and observed the outcome of their responses, and thus are not grounded in the world. SayCan grounds LLMs via value functions of pretrained skills, allowing them to execute real-world, abstract, long-horizon commands on robots.

1 Introduction

Recent progress in training large language models (LLMs) has led to systems that can generate complex text based on prompts, answer questions, or even engage in dialogue on a wide range of topics. These models absorb vast quantities of knowledge from text corpora mined from the web, and we might wonder whether knowledge of everyday tasks that is encoded in such models can be used by robots to perform complex tasks in the real world. But how can embodied agents extract and harness the knowledge of LLMs for physically grounded tasks?

This question poses a major challenge. LLMs are not grounded in the physical world and they do not observe the consequences of their generations on any physical process [1]. This can lead

[1] We plan to open source a toy domain discussed in Appendix Section E and Figure 16.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

LLMs to not only make mistakes that seem unreasonable or humorous to people, but also to interpret instructions in ways that are nonsensical or unsafe for a particular physical situation. Figure 1 shows an example – a kitchen robot capable of executing skills such as "pick up the sponge" or "go to the table" may be asked for help cleaning up a spill ("I spilled my drink, can you help?"). A language model may respond with a reasonable narrative that is not feasible or useful for the robot. "You could try using a vacuum cleaner" is impossible if there is no vacuum in the scene or if the robot is incapable of using one. With prompt engineering, a LLM may be capable of splitting the high-level instruction into sub-tasks, but it cannot do so without the context of what the robot is capable of given its abilities *and* the current state of the robot *and* the environment.

Motivated by this example, we study the problem of how to extract the knowledge in LLMs for enabling an embodied agent, such as a robot, to follow high-level textual instructions. The robot is equipped with a repertoire of learned skills for "atomic" behaviors that are capable of low-level visuomotor control. We make use of the fact that, in addition to asking the LLM to simply interpret an instruction, we can use it to score the likelihood that an individual skill makes progress towards completing the high-level instruction. Then, if each skill has an affordance function that quantifies how likely it is to succeed from the current state (such as a learned value function), its value can be used to weight the skill's likelihood. In this way, the LLM describes the probability that each skill contributes to completing the instruction, and the affordance function describes the probability that each skill will succeed – combining the two provides the probability that each skill will perform the instruction *successfully*. The affordance functions make the LLM aware of the current scene, and constraining the completions to the skill descriptions makes the LLM aware of the robot's capabilities. Furthermore, this combination results in a fully explainable sequence of steps that the robot will execute to accomplish an instruction – an interpretable plan that is expressed through language.

Our method, SayCan, extracts and leverages the knowledge within LLMs in physically-grounded tasks. The LLM (Say) provides a task-grounding to determine useful actions for a high-level goal and the learned affordance functions (Can) provide a world-grounding to determine what is possible to execute upon the plan. We use reinforcement learning (RL) as a way to learn language-conditioned value functions that provide affordances of what is possible in the world. We evaluate the proposed approach on about 100 real-world robotic tasks that involve a mobile robot accomplishing a large set of language instructions in a real kitchen in a zero-shot fashion. Our experiments validate that SayCan can execute temporally-extended, abstract instructions. Grounding the LLM in the real-world via affordances nearly doubles the performance over the non-grounded baselines.

2 SayCan: Do As I Can, Not As I Say

Problem Statement. Our system receives a user-provided natural language instruction i that describes a task that the robot should execute. The instruction can be long, abstract, or ambiguous. We also assume that we are given a set of skills Π, where each skill $\pi \in \Pi$ performs a short task, such as picking up a particular object, and comes with a short language description ℓ_π (e.g., "find a sponge") and an affordance function $p(c_\pi|s, \ell_\pi)$, which indicates the probability of c-ompleting the skill with description ℓ_π successfully from state s. Intuitively, $p(c_\pi|s, \ell_\pi)$ means "if I ask the robot to do ℓ_π, will it do it?". In RL terminology, $p(c_\pi|s, \ell_\pi)$ is the value function for the skill if we take the reward to be 1 for successful completion and 0 otherwise.

As mentioned above, ℓ_π denotes the textual label of skill π and $p(c_\pi|s, \ell_\pi)$ denotes the probability that skill π with textual label ℓ_π successfully completes if executed from state s, where c_π is a Bernoulli random variable. The LLM provides us with $p(\ell_\pi|i)$, the probability that a skill's textual label is a valid next step for the user's instruction. However, what we are interested in is the probability that a given skill successfully makes progress toward actually completing the instruction, which we denote as $p(c_i|i, s, \ell_\pi)$. Assuming that a skill that succeeds makes progress on i with probability $p(\ell_\pi|i)$ (i.e., its probability of being the right skill), and a skill that fails makes progress with probability zero, we can factorize this as $p(c_i|i, s, \ell_\pi) \propto p(c_\pi|s, \ell_\pi)p(\ell_\pi|i)$. This corresponds to multiplying the probability of the language description of the skill given the instruction $p(\ell_\pi|i)$, which we refer to as task-grounding, and the probability of the skill being possible in the current state of the world $p(c_\pi|s, \ell_\pi)$, which we refer to as world-grounding.

Connecting Large Language Models to Robots. While large language models can draw on a wealth of knowledge learned from copious amounts of text, they will not necessarily break down high-level commands into low-level instructions that are suitable for robotic execution. If a language model were asked "how would a robot bring me an apple", it may respond "a robot could go to a

nearby store and purchase an apple for you". Though this response is a reasonable completion for the prompt, it is not necessarily actionable to an embodied agent, which may have a narrow and fixed set of abilities. Therefore, to adapt language models to our problem statement, we must somehow inform them that we specifically want the high-level instruction to be broken down into sequences of available low-level skills. One approach is careful prompt engineering [2, 3], a technique to coax a language model to a specific response structure. Prompt engineering provides examples in the context text ("prompt") for the model that specify the task and the response structure which the model will emulate; the prompt used in this work is shown in Appendix C.4 along with experiments ablating it. However, this is not enough to fully constrain the output to admissible primitive skills for an embodied agent, and indeed at times it can produce inadmissible actions or language that is not formatted in a way that is easy to parse into individual steps.

Scoring language models open an avenue to constrained responses by outputting the probabilities assigned by a language model to fixed outputs. A language model represents a *distribution* over potential completions $p(w_k|w_{<k})$, where w_k is a word that appears at a kth position in a text. While typical generation applications (e.g., conversational agents) sample from this distribution or decode the maximum likelihood completion, we can also use the model to *score* a candidate completion selected from a set of options. Formally in SayCan, given a set of low-level skills Π, their language descriptions ℓ_Π and an instruction i, we compute the probability of a language description of a skill $\ell_\pi \in \ell_\Pi$ making progress towards executing the instruction i: $p(\ell_\pi|i)$, which corresponds to querying the model over potential completions. The optimal skill according to the language model is computed via $\ell_\pi = \arg\max_{\ell_\pi \in \ell_\Pi} p(\ell_\pi|i)$. Once selected, the process proceeds by iteratively selecting a skill and appending it to the instruction. Practically, in this work we structure the planning as a dialog between a user and a robot, in which a user provides the high level-instruction (e.g., "How would you bring me a coke can?") and the language model responds with an explicit sequence ("I would: 1. ℓ_π", e.g., "I would: 1. find a coke can, 2. pick up the coke can, 3. bring it to you").

This has the added benefit of interpretability, as the model not only outputs generative responses, but also gives a notion of likelihood across many possible responses. Figure 2 (and Appendix Figure 9 in more detail) shows this process of forcing the LLM into a language pattern, where the set of tasks are the skills the low-level policy is capable of and prompt engineering shows plan examples and dialog between the user and the robot. With this approach, we are able to effectively extract knowledge from the language model, but it leaves a major issue: while the decoding of the instruction obtained in this way always consists of skills that are available to the robot, these skills may not necessarily be appropriate for executing the desired high-level task in the *specific* situation that the robot is currently in. For example, if I ask a robot to "bring me an apple", the optimal set of skills changes if there is no apple in view or if it already has one in its hand.

SayCan. The key idea of SayCan is to ground large language models through value functions – affordance functions that capture the log likelihood that a particular skill will be able to succeed in the current state. Given a skill $\pi \in \Pi$, its language description ℓ_π and its corresponding value function, which provides $p(c_\pi|s, \ell_\pi)$, the probability of c-ompletion for the skill described by ℓ_π in state s, we form an affordance space $\{p(c_\pi|s, \ell_\pi)\}_{\pi \in \Pi}$. This value function space captures affordances across all skills [4]. For each skill, the affordance function and the LLM probability are then multiplied together and ultimately the most probable skill is selected, i.e. $\pi = \arg\max_{\pi \in \Pi} p(c_\pi|s, \ell_\pi) p(\ell_\pi|i)$. Once the skill is selected, the corresponding policy is executed by the agent and the LLM query is amended to include ℓ_π and the process is run again until a termination token (e.g., "done") is chosen. This process is shown in Figure 2 and described in Appendix Algorithm 1. These two mirrored processes together lead to a probabilistic interpretation of SayCan, where the LLM provides probabilities of a skill being useful for the high-level instruction and the affordances provide probabilities of successfully executing each skill. Combining these two probabilities together provides a probability that this skill furthers the execution of the high-level instruction commanded by the user.

3 Implementing SayCan in a Robotic System

Language-Conditioned Robotic Control Policies. To instantiate SayCan, we must provide it with a set of skills, each of which has a policy, a value function, and a short language description (e.g., "pick up the can"). These skills, value functions, and descriptions can be obtained in a variety of different ways. In our implementation, we train the individual skills either with image-based behavioral cloning, following the BC-Z method [5], or reinforcement learning, following MT-Opt [6]. Regardless of how the skill's policy is obtained, we utilize value functions trained via TD backups

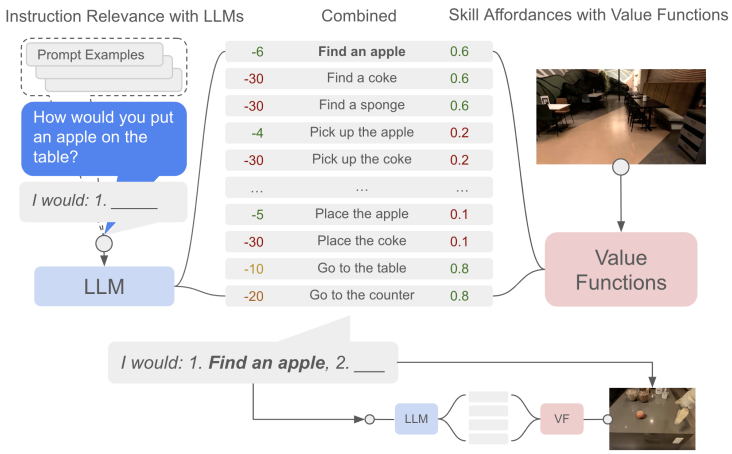


Figure 2: Given a high-level instruction, SayCan combines probabilities from a LLM (the probability that a skill is useful for the instruction) with the probabilities from a value function (the probability of successfully executing said skill) to select the skill to perform. This emits a skill that is both possible and useful. The process is repeated by appending the skill to the response and querying the models again, until the output step is to terminate. Appendix Figures 9 and 10 focus on the LLM and VFS components.

as the affordance model for that skill. While we find that the BC policies achieve higher success rates at the current stage of our data collection process, the value functions provided by the RL policies are crucial as an abstraction to translate control capabilities to a semantic understanding of the scene. In order to amortize the cost of training many skills, we utilize multi-task BC and multi-task RL, respectively, where instead of training a separate policy and value function per skill, we train multi-task policies and models that are *conditioned* on the language description. Note, however, that this description only corresponds to *low level* skills – it is still the role of the LLM in SayCan to interpret the high-level instruction and break it up into individual low level skill descriptions.

To condition the policies on language, we utilize a pre-trained large sentence encoder language model [7]. We freeze the language model parameters during training and use the embeddings generated by passing in text descriptions of each skill. These text embeddings are used as the input to the policy and value function that specify which skill should be performed (see the details of the architectures used in the Appendix B.1). Since the language model used to generate the text embeddings is not necessarily the same as the language model used for planning, SayCan is able to utilize different language models well suited for different abstraction levels – understanding planning with respect to many skills as opposed to expressing specific skills more granularly.

Training the Low-Level Skills. We utilize both BC and RL policy training procedures to obtain the language-conditioned policies and value functions, respectively. To complete the description of the underlying MDP that we consider, we provide the reward function as well as the skill specification that is used by the policies and value functions. As mentioned previously, for skill specification we use a set of short, natural language descriptions that are represented as language model embeddings. We utilize sparse reward functions with reward values of 1.0 at the end of an episode if the language command was executed successfully, and 0.0 otherwise. The success of language command execution is rated by humans where the raters are given a video of the robot performing the skill, together with the given instruction. If two out of the three raters agree that the skill was accomplished successfully, the episode is labelled with a positive reward.

To learn language-conditioned BC policies at scale in the real world, we build on top of BC-Z [5] and use a similar policy-network architecture (shown in Fig. 7). To learn a language-conditioned RL policy, we use MT-Opt [6] in a simulator using RetinaGAN sim-to-real transfer [8]. We bootstrap the performance of simulation policies by utilizing simulation demonstrations to provide initial successes, and then continuously improve the RL performance with online data collection. We use a network architecture similar to MT-Opt (shown in Fig. 6). The action space of our policies includes the six degrees of freedom of the end-effector pose as well as gripper open and close commands, x-y position and yaw orientation delta of the mobile base of the robot, and the *terminate* action. Additional details on data collection and training are in Appendix Section B.2.

Robotic System and Skills. For the control policies, we study a diverse set of manipulation and navigation skills using a mobile manipulator robot. Inspired by common skills one might pose to a robot in a kitchen environment, we propose 551 skills that span seven skill families and 17 objects, which include picking, placing and rearranging objects, opening and closing drawers, navigating to various locations, and placing objects in a specific configurations. In this study we utilize the skills that are most amenable to more complex behaviors via composition and planning as well as those that have high performance at the current stage of data collection; for more details, see Appendix C.

4 Experimental Evaluation

Figure 3: The experiments were performed in an office kitchen and a mock kitchen mirroring this setup, with 5 locations and 15 objects. The robot is a mobile manipulator with policies trained from an RGB observation.

Experimental Setup We evaluate SayCan with a mobile manipulator and a set of object manipulation and navigation skills in two office kitchen environments. Figure 3 shows the environment setup and the robot. We use 15 objects commonly found in an office kitchen and 5 known locations with semantic meaning (two counters, a table, a trash can, and the user location). We test our method in two environments: a real office kitchen and a mock environment mirroring the kitchen, which is also the environment in which the robot's skills were trained. The robot used is a mobile manipulator with a 7 degree-of-freedom arm and a two-fingered gripper. The LLM used is 540B PaLM [9].

Instructions. To evaluate SayCan, we test across 101 instructions from 7 instruction families, summarized in Table 1 and enumerated in Appendix D.1. These were developed to test various aspects of SayCan and were inspired by crowd sourcing via Amazon Mechanical Turk and in-person kitchen users, as well as benchmarks such as ALFRED [10] and BEHAVIOR [11]. The instructions span multiple axes of variation: time-horizon (from single primitives to 10+ in a row), language complexity (from structured language to fully crowd-sourced requests), and embodiment (variations over the robot and environment state). Table 1 details examples for each family.

Instruction Family	Num	Explanation	Example Instruction
NL Single Primitive	15	NL queries for a single primitive	Let go of the coke can
NL Nouns	15	NL queries focused on abstract nouns	Bring me a fruit
NL Verbs	15	NL queries focused on abstract verbs	Restock the rice chips on the far counter
Structured Language	15	Structured language queries, mirror NL Verbs	Move the rice chips to the far counter.
Embodiment	11	Queries to test SayCan's understanding of the current state of the environment and robot	Put the coke on the counter. (starting from different completion stages)
Crowd-Sourced	15	Queries in unstructured formats	My favorite drink is redbull, bring one
Long-Horizon	15	Long-horizon queries that require many steps of reasoning	I spilled my coke on the table, throw it away and bring me something to clean

Table 1: **List of instruction family definitions:** We evaluate the algorithm on 101 instructions. We group the instructions into different families, with each family focusing on testing one aspect of the proposed method.

Metrics. To understand the performance of the proposed method we measure two main metrics. The first is **plan success rate**, which measures whether the skills selected by the model are correct for the instruction, regardless of whether or not they actually successfully executed. We ask 3 human raters to indicate whether the plan generated by the model can achieve the instruction, and if 2 out of 3 raters agree that the plan is valid, it is marked a success. Note that for many instructions there may be multiple valid solutions. For example if the instruction is to "bring a sponge and throw away the soda can", the plan can choose to bring sponge first or throw away the soda can first.

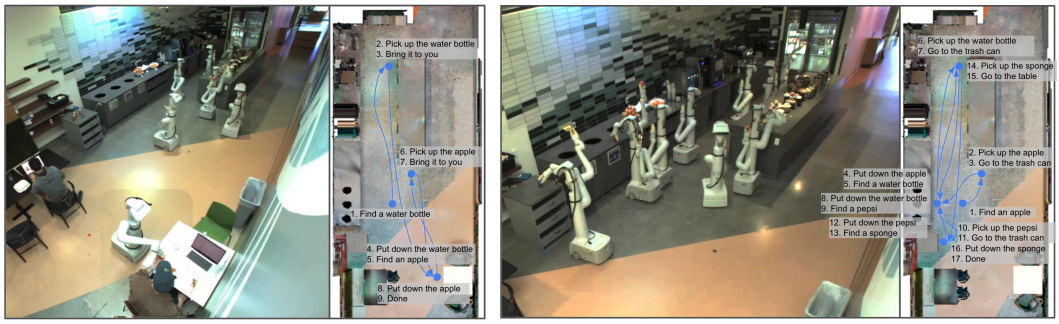
The second metric is **execution success rate**, which measures whether the full SayCan system actually performs the desired instruction successfully. We ask 3 human raters to watch the robot execution. The raters are asked to answer the question "whether the robot achieves the task specified by the task string?" We mark an execution successful if 2 out of 3 raters agree that it is successful.

4.1 Results

Table 2 shows the performance of SayCan across 101 tasks. In the mock kitchen, SayCan achieved a planning success rate of 84% and an execution rate of 74%. We also investigate SayCan out of the lab setting and in the real kitchen to verify the performance of the policies and value functions in

this setting. We find a reduction of planning performance by 3% and execution by 14%, indicating SayCan and the underlying policies generalize reasonably well to the full kitchen. The full task list and results can be found in the Appendix Table 3, LLM size ablations can be found in Appendix C.3, and videos of experiment rollouts and the decision making process can be found in the supplemental.

Figure 4 shows two long-horizon queries and the resulting rollouts. These tasks require SayCan to plan many steps without error and for the robot to navigate and interact with a significant portion of the kitchen. Each query requires SayCan to understand context implicit within the instruction. In Figure 4a, the algorithm must understand the operator has asked for something to "recover from a workout", i.e. something healthy, and thus it brings water and an apple rather than, e.g., a soda and chips. Furthermore, the algorithm must understand ordering and history, that it has already brought a drink and now must bring a snack before terminating. In Figure 4b, SayCan must track which objects are the "them" that need to be disposed of and where the sponge should be brought.



(a) "I just worked out, can you bring me a drink and a snack to recover?"

(b) "I left out a coke, apple, and water, can you throw them away and then bring me a sponge to wipe the table?"

Figure 4: Timelapse of rollouts to two long-horizon queries. The robot interacts with a large portion of the kitchen environment and successfully performs sequences of manipulation and navigation skills.

Figure 5 highlights SayCan's decision making, along with its interpretability. The decision making process can be understood as it solves instructions by visualizing what the two sides of the algorithm output. This allows a user to understand what options SayCan is considering as language completions and what it believes is possible. We find that sequence order is understood (approaching objects before picking them up and picking them up before bringing them). Figure 5 shows that though the query mentions a coke, SayCan understands that the important object is something to clean and brings a sponge. Appendix D.5 shows additional rollouts with complex decisions, embodiment grounding, and long-horizon tasks in Figures 12-15 as well as failures in Figure 14. We believe such real-time and clear interpretability opens avenues to more interactive operation.

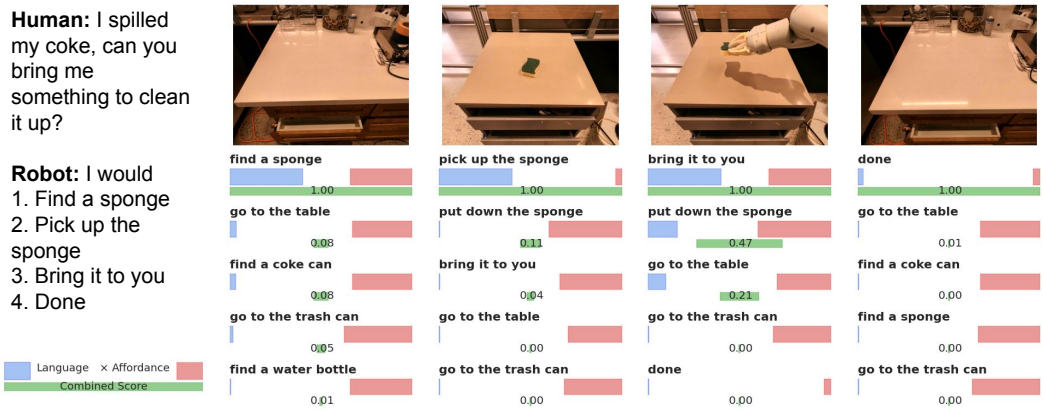


Figure 5: Visualization of SayCan's decision making, where the top combined score chooses the correct skill.

When comparing the performance of different instruction families in Table 2 (see Table 1 for an explanation of families), we see that the natural language nouns performed worse than natural lan-

guage verbs, due to the number of nouns possible (15 objects and 5 locations) versus number of verbs (6). The structured language tasks (created to ablate the performance loss of spelling out the solution versus understanding the query) were planned correctly 93% of the time, while their natural language verb counterparts were planned correctly 100%. This indicates the language model effectively parses the queries. The embodiment tasks were planned correctly 64% of the time, generally with failures as a result of affordance function misclassification. SayCan planned and executed crowd-sourced natural queries with performance on par with other instruction families. SayCan performed worst on the most challenging long-horizon tasks, where most failures were a result of early termination by the LLM (e.g., bringing one object but not the second). We also find that SayCan struggles with negation (e.g., "bring me a snack that isn't an apple") and ambiguous references (e.g. asking for drinks with caffeine), which is a known issue inherited from underlying language models [12]. Overall, 65% of the errors were LLM failures and 35% were affordance failures.

Returning to our initial example, "I spilled something, can you help?", an ungrounded language model would respond with statements like "I can call you a cleaner" or "I can vacuum that up for you", which given our robot are unreasonable. We have shown that SayCan responds "I would: 1. find a sponge, 2. pick up the sponge, 3. bring it to you, 4. done" and is able execute this sequence on the robot in a real kitchen. This requires long-horizon reasoning over a required order, an abstract understanding of the instruction, and knowledge of both the environment and robot's capabilities.

Ablating Language. To study the importance of the LLM, we conduct two ablation experiments using the language-conditioned policy (see Sections 3-3). In *BC NL* we feed the full instruction i into the policy – this approach is representative of standard RL or BC-based instruction following methods [5, 13, 14, 15]. In *BC USE* we project the high-level instruction into the set of known language commands via the Universal Sentence Encoder (USE) embeddings [7] by embedding the instruction, all the tasks, and the combinatorial set of sequences tasks (i.e., we consider "pick coke can" as well as "1. find coke can, 2. pick coke can" and so on), and selecting the highest cosine similarity instruction. The results in Table 2 illustrate the necessity of the language grounding where *BC NL* achieves 0% in all tasks and *BC USE* achieves 60% for single primivites, but 0% otherwise.

Ablating Value Functions. Table 2 illustrates the necessity of the affordance grounding. We compare SayCan to (1) *No VF*, which removes the value function grounding (i.e., choosing the maximum language score skill) and to (2) *Generative*, which uses the generative output of the LLM and then projects each planned skill to its maximal cosine similarity skill via USE embeddings. The latter in effect compares to [16], which loses the explicit option probabilities, and thus is less interpretable and cannot be combined with affordance probabilities. For *Generative* we also tried BERT embeddings [17], but found poor performance. The *No VF* and *Generative* approaches performed similarly, achieving 67% and 74% planning success rate respectively, and worse than SayCan's 84%.

4.2 Case Study of New Capabilities

Adding Skills: Drawer Manipulation (Appendix D.2). SayCan is capable of integrating new skills by simply adding the new skills as options for the LLM and provide accompanying value functions and add an example in the prompt with that skill. For example, with the skills open, close, and go to the drawer, SayCan is capable of solving tasks such as "restock the coke and pepsi into the drawer". Over 21 queries we found a planning rate of 100% and an execution rate of 33% (due to failures of the chained manipulation policy), with no loss in performance for other instructions.

Chain of Thought Reasoning (Appendix D.3). SayCan can be integrated with recent work improving LLM reasoning, such as Chain of Thought [18].

Multilingual Queries (Appendix D.4). The LLM was trained on a multilingual corpus and thus SayCan can handle multilingual queries with comparable success rate as English.

5 Related Work

Grounding Language Models. A significant body of work has focused on grounding language [19, 20]. Recent works have studied how to ground modern language models, by training them to accept additional environment inputs [21, 22, 23, 24, 25] or to directly output actions [26, 27, 28]. Others grounded language in an environment through prompt engineering [18]. Concurrently with SayCan, Huang et al. [16] use prompt engineering to extract temporally extended plans, but without any additional mechanism to ensure grounding, roughly corresponding to the "Generative" baseline in our experiments. The above methods are all trained without interaction with a physical environment,

		Mock Kitchen		Kitchen		No Affordance		No LLM	
		SayCan	**SayCan**	**SayCan**	**SayCan**	**No VF**	**Gen.**	**BC NL**	**BC USE**
Family	**Num**	Plan	Execute	Plan	Execute	Plan	Plan	Execute	Execute
NL Single	15	100%	100%	93%	87%	73%	87%	0%	60%
NL Nouns	15	67%	47%	60%	40%	53%	53%	0%	0%
NL Verbs	15	100%	93%	93%	73%	87%	93%	0%	0%
Structured	15	93%	87%	93%	47%	93%	100%	0%	0%
Embodiment	11	64%	55%	64%	55%	18%	36%	0%	0%
Crowd Sourced	15	87%	87%	73%	60%	67%	80%	0%	0%
Long-Horizon	15	73%	47%	73%	47%	67%	60%	0%	0%
Total	101	84%	74%	81%	60%	67%	74%	0%	9%

Table 2: Success rates of instructions by family. SayCan achieves a planning success rate of 84% and execution success rate of 74% in the training environment and 81% planning and 60% execution in a real kitchen. *No VF* uses the maximum score skill from the LLM, *Generative (Gen.)* uses a generative LLM and then projects to the nearest skill via USE embeddings, *BC NL* uses the policy with the natural language instruction, and *BC USE* uses the policy with the natural language instruction projected to the nearest skill via USE embeddings.

thus limiting their ability to reason over embodied interactions. One approach to grounding language models in interaction is by learning downstream networks with pre-trained LLM representations [29, 15, 14, 30, 31, 32, 33, 34]. Another approach finetunes language models with interactive data, such as rewards or ranking feedback of the interaction [3, 35, 36]. In our work, SayCan is able to ground language models in the given environment through previously-trained value functions, enabling general, long-horizon behaviors in a zero-shot manner, i.e., without additional training.

Learning Language-Conditioned Behavior. There is a long history of research studying how to connect language and behavior [37, 38, 39, 40, 41]. A large number of prior works have learned language-conditioned behavior via imitation learning [42, 15, 13, 5, 43, 28] or reinforcement learning [44, 45, 40, 46, 47, 48, 14, 32]. Most of these prior works focus on following low-level instructions, such as for pick-and-place tasks and other robotic manipulation primitives [13, 15, 48, 14, 5, 43], though some methods address long-horizon, compound tasks in simulated domains [49, 50, 46]. Like these latter works, we focus on completing temporally extended tasks. However, a central aspect of our work is to solve such tasks by extracting and leveraging the knowledge in large language models. While prior works have studied how pre-trained language embeddings can improve generalization to new instructions [29, 15, 14] and to new low-level tasks [5], we extract much more substantial knowledge from LLMs by grounding them within the robot's affordances. This allows robots to use language models for planning.

Task Planning and Motion Planning. Task and motion planning [51, 52] is a problem of sequencing tasks to solve a high-level problem, while ensuring the feasibility given an embodiment (task planning [53, 54, 55]; motion planning [56]). Classically, this problem has been solved through symbolic planning [53, 55] or optimization [57, 58], but these require explicit primitives and constraints. Machine learning has recently been applied to enable abstract task specification, allow general primitives, or relax constraints [59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]. Others learn to hierarchically solve such long-horizon problems [71, 72, 4, 73, 46]. SayCan leverages an LLM's semantic knowledge about the world for interpreting instructions *and* understanding how to execute them. The use of LLMs and generality of learned low-level policies enables long-horizon, abstract tasks that scale effectively to the real world, as demonstrated in our robot experiments.

6 Conclusions and Limitations

We presented SayCan, a method that enables leveraging and grounding the rich knowledge in large language models to complete embodied tasks. For real-world grounding, we use RL-based value functions for the individual skills that provide affordances of what is possible in the world, and use textual labels for these skills as potential responses that are scored by a language model. We evaluated this approach on a number of real-world robotic tasks that involve a mobile manipulator robot accomplishing a large set of long-horizon natural language instructions in a real kitchen.

While SayCan presents a viable way to ground language models in agents' affordances, it has a number of limitations. First, we expect this method to inherit the limitations and biases of LLMs [74, 75], including the dependence on the training data. In addition, we observe that even though SayCan allows the users to interact with the agents using natural language commands, the primary bottleneck of the system is in the range and capabilities of the underlying skills. To illustrate this, we present representative failure cases in Appendix D.

References

[1] E. M. Bender and A. Koller. Climbing towards nlu: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Preprint*, 2022.

[4] D. Shah, P. Xu, Y. Lu, T. Xiao, A. Toshev, S. Levine, and B. Ichter. Value function spaces: Skill-centric state abstractions for long-horizon reasoning. *ICLR*, 2022. URL https://arxiv.org/pdf/2111.03189.pdf.

[5] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2021.

[6] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv*, 2021.

[7] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[8] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926, 2021.

[9] A. Chowdhery, S. Narang, J. Devlin, et al. Palm: Scaling language modeling with pathways. 2022. URL https://storage.googleapis.com/pathways-language-model/PaLM-paper.pdf.

[10] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[11] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR, 2022.

[12] A. Hosseini, S. Reddy, D. Bahdanau, R. D. Hjelm, A. Sordoni, and A. Courville. Understanding by understanding not: Modeling negation in language models. *arXiv preprint arXiv:2105.03519*, 2021.

[13] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor. Language-conditioned imitation learning for robot manipulation tasks. *ArXiv*, abs/2010.12083, 2020.

[14] S. Nair, E. Mitchell, K. Chen, B. Ichter, S. Savarese, and C. Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2021.

[15] C. Lynch and P. Sermanet. Grounding language in play. 2020.

[16] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[19] J. M. Siskind. Grounding language in perception. *Artificial Intelligence Review*, 1994.

[20] T. Winograd. Understanding natural language. *Cognitive psychology*, 1972.

[21] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[22] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[23] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 2019.

[24] R. Zellers, X. Lu, J. Hessel, Y. Yu, J. S. Park, J. Cao, A. Farhadi, and Y. Choi. Merlot: Multimodal neural script knowledge models. *Advances in Neural Information Processing Systems*, 2021.

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[26] A. Suglia, Q. Gao, J. Thomason, G. Thattai, and G. Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *arXiv preprint arXiv:2108.04927*, 2021.

[27] A. Pashevich, C. Schmid, and C. Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[28] P. Sharma, A. Torralba, and J. Andreas. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021.

[29] F. Hill, S. Mokra, N. Wong, and T. Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*, 2020.

[30] V. Blukis, C. Paxton, D. Fox, A. Garg, and Y. Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. In *Conference on Robot Learning*, 2022.

[31] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[32] A. Akakzia, C. Colas, P.-Y. Oudeyer, M. Chetouani, and O. Sigaud. Grounding language to autonomously-acquired skills via goal generation. *arXiv preprint arXiv:2006.07185*, 2020.

[33] R. Zellers, A. Holtzman, M. Peters, R. Mottaghi, A. Kembhavi, A. Farhadi, and Y. Choi. Piglet: Language grounding through neuro-symbolic interaction in a 3d world. *arXiv preprint arXiv:2106.00188*, 2021.

[34] P. C. Humphreys, D. Raposo, T. Pohlen, G. Thornton, R. Chhaparia, A. Muldal, J. Abramson, P. Georgiev, A. Goldin, A. Santoro, et al. A data-driven approach for learning to control computers. *arXiv preprint arXiv:2202.08137*, 2022.

[35] M. Reid, Y. Yamada, and S. S. Gu. Can wikipedia help offline reinforcement learning. *arXiv preprint arXiv:2201.12122*, 2022.

[36] S. Li, X. Puig, Y. Du, C. Wang, E. Akyurek, A. Torralba, J. Andreas, and I. Mordatch. Pretrained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

[37] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. 01 2006.

[38] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *HRI 2010*, 2010.

[39] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. volume 2, 01 2011.

[40] J. Luketina, N. Nardelli, G. Farquhar, J. N. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. In *IJCAI*, 2019.

[41] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 2020.

[42] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, 2016.

[43] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, 2022.

[44] D. K. Misra, J. Langford, and Y. Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP*, 2017.

[45] K. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. Czarnecki, M. Jaderberg, D. Teplyashin, M. Wainwright, C. Apps, D. Hassabis, and P. Blunsom. Grounded language learning in a simulated 3d world. *ArXiv*, abs/1706.06551, 2017.

[46] Y. Jiang, S. Gu, K. Murphy, and C. Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *NeurIPS*, 2019.

[47] G. Cideron, M. Seurin, F. Strub, and O. Pietquin. Self-educated language agent with hindsight experience replay for instruction following. *ArXiv*, abs/1910.09451, 2019.

[48] P. Goyal, S. Niekum, and R. Mooney. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. *ArXiv*, abs/2007.15543, 2020.

[49] J. Oh, S. Singh, H. Lee, and P. Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. *ArXiv*, abs/1706.05064, 2017.

[50] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. *ArXiv*, abs/1611.01796, 2017.

[51] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[52] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, 2014.

[53] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 1971.

[54] E. D. Sacerdoti. A structure for plans and behavior. Technical report, SRI International, Menlo Park California Artificial Intelligence Center, 1975.

[55] D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila. Shop: Simple hierarchical ordered planner. 1999.

[56] S. M. LaValle. *Planning algorithms*. 2006.

[57] M. Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[58] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018.

[59] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[60] D. Xu, R. Martín-Martín, D.-A. Huang, Y. Zhu, S. Savarese, and L. F. Fei-Fei. Regression planning networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[61] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[62] B. Eysenbach, R. R. Salakhutdinov, and S. Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 2019.

[63] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

[64] B. Ichter, P. Sermanet, and C. Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. *Conference on Robot Learning (CoRL)*, 2021.

[65] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A joint model of language and perception for grounded attribute learning. *arXiv preprint arXiv:1206.6423*, 2012.

[66] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Perez, L. P. Kaelbling, and J. Tenenbaum. Inventing relational state and action abstractions for effective and efficient bilevel planning. *arXiv preprint arXiv:2203.09634*, 2022.

[67] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox. Online replanning in belief space for partially observable task and motion problems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[68] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[69] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 2016.

[70] B. Wu, S. Nair, L. Fei-Fei, and C. Finn. Example-driven model-based reinforcement learning for solving long-horizon visuomotor tasks. In *5th Annual Conference on Robot Learning*, 2021.

[71] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *ArXiv*, abs/1909.05829, 2020.

[72] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese. Relmogen: Integrating motion generation in reinforcement learning for mobile manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[73] C. Li, F. Xia, R. Martin-Martin, and S. Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, 2020.

[74] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[75] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[76] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[77] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, pages 1–67, 2019.

[78] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[79] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[80] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[81] J. J. Gibson. The theory of affordances. *The Ecological Approach to Visual Perception*, 1977.

[82] T. Xiao, E. Jang, D. Kalashnikov, S. Levine, J. Ibarz, K. Hausman, and A. Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. In *International Conference on Learning Representations*, 2019.

[83] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *ICLR*, 2016.

[84] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921*, 2021.