LANGUAGE MODELS AS ZERO-SHOT PLANNERS: EXTRACTING ACTIONABLE KNOWLEDGE FOR EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Can world knowledge learned by large language models (LLMs) be used to act in interactive environments? In this paper, we investigate the possibility of grounding high-level tasks, expressed in natural language (i.e. "make breakfast"), to a chosen set of actionable steps (i.e. "open fridge"). While prior work focused on learning from explicit step-by-step examples of how to act, we surprisingly find that if pre-trained LMs are large enough and prompted appropriately, they can effectively decompose high-level tasks into low-level plans without any further training. However, the plans produced naively by LLMs often cannot map precisely to admissible actions. We propose a procedure that conditions on existing demonstrations and semantically translates the plans to admissible actions. Our evaluation in the recent VirtualHome environment shows that the resulting method substantially improves executability over the LLM baseline. The conducted human evaluation reveals a trade-off between executability and correctness but shows a promising sign towards extracting actionable knowledge from language models[1].

1 INTRODUCTION

Large language models (LLMs) have made impressive advances in language generation and understanding in recent years (Devlin et al., 2018; Radford et al., 2019; Raffel et al., 2019; Brown et al., 2020). See Bommasani et al. (2021) for a recent summary of their capabilities and impacts. Being trained on large corpora of human-produced language, these models are thought to contain a lot of information about the world (Roberts et al., 2020; Li et al., 2021; BIG-bench collaboration, 2021) - albeit in linguistic form.

We ask whether we can use such knowledge contained in LLMs not just for linguistic tasks, but to make goal-driven decisions to that can be enacted in interactive, embodied environments. But we are not simply interested in whether we can train models on a dataset of demonstrations collected for some specific environment – we are instead interested in whether LLMs *already contain* information necessary to accomplish goals without any additional training.

More specifically, we ask whether world knowledge about how to perform high-level tasks (such as "make breakfast") can be expanded to a series of groundable actions (such as "open fridge", "grab milk", "close fridge", etc) that can be executed in the environment. For our investigation, we use recently proposed VirtualHome environment (Puig et al., 2018). It can simulate a large variety of realistic human activities in a household environment and supports ability to perform them via embodied actions defined with a `verb-object` syntax. However, due to open-ended nature of the tasks, it is difficult to autonomously evaluate their success. We rely on human evaluation (conducted on Mechanical Turk) to decide whether sequences of actions meaningfully accomplish posed tasks.

We find that large GPT-3 (Brown et al., 2020) and Codex (Chen et al., 2021) models, when prompted with a single fixed example of a task description and its associated sequence of actions, can produce very plausible action plans for the task we're interested in. Such completions reflect the information already stored in the model – no model fine-tuning is involved. Additionally, we only observe this

[1]Results and videos at `https://sites.google.com/view/language-model-as-planner`

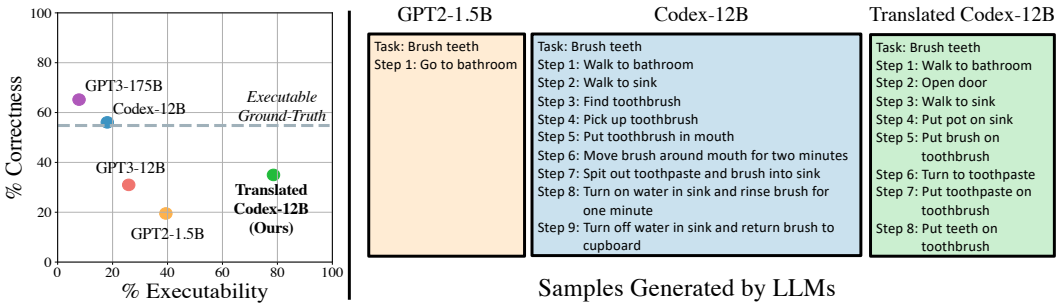


Figure 1: Executability v.s. semantic correctness of generated action plans **(left)** and sample action plans generated by different models **(right)**. Large models can produce action plans indistinguishable from plans created by humans, but frequently are not executable in the environment. Using our techniques, we can significantly improve executability, albeit at the cost of correctness.

effect in the larger models. Unfortunately, despite their semantic correctness, the produced action plans are often not executable in the environment. Produced actions may not map precisely to admissible actions, or may contain various linguistic ambiguities.

We propose several tools to improve executability of the model's outputs. First, we enumerate all admissible action phrases and map the model's output action to the most semantically-similar admissible action (we use similarity measure between sentence embeddings produced by a RoBERTa model Liu et al. (2019) in this work, but other choices are possible). Second, we use the model to autoregressively generate actions in a plan by conditioning past actions that have been made admissible via the technique above. Such on the fly correction can keep generation anchored to admissible actions. Third, we provide weak supervision to the model by prompting the model with a known task example similar to the query task. This is somewhat reminiscent of prompt tuning approaches, but does not require access to gradients or internals of the model.

Using above tools to bias model generation, we find that we improve executability of instructions from 18% to 79% (see Figure 1) without any invasive modifications to model parameters or any extra gradient or internal information beyond what is returned from the model's forward pass. This is advantageous because it does not require any modifications to model training procedure and can fit within existing model serving pipelines. However, we do find there to be a significant drop in correctness of the instruction sequences generated with above tools (as judged by humans), indicating a promising step, but requiring more research on the topic.

To summarize, our paper's contributions are as follows:

- We show that without any training, large language models can be prompted to generate plausible goal-driven action plans, but such plans are frequently not executable in interactive environments.
- We propose several tools to improve executability of the model generation without invasive probing or modifications to the model.
- We conduct a human evaluation of multiple techniques and models and report on the tradeoffs between executabiltiy and semantic correctness.

2 Evaluation Framework

Simulating open-ended tasks that resemble naturalistic human activities requires an environment to support a rich set of diverse interactions, rendering most existing embodied environments unsuitable for our investigation. One exception is VirtualHome (Puig et al., 2018), which models human activities in a typical household. Therefore, we only provide evaluation in this environment. To further measure correctness given open-ended tasks, we conduct a human evaluation. We note that since no further training is involved throughout our investigations, the observations and findings presented in this paper should also translate to similar embodied environments.

2.1 EVALUATED ENVIRONMENT: VIRTUALHOME

Preliminaries In VirtualHome, activities are expressed as programs. Each program consists of a sequence of steps, where each step is written as: $[action]$ $\langle arg_1 \rangle (id_1)$ ... $\langle arg_n \rangle (id_n)$. Each $action$ refers to atomic actions such as "walk", "open", and "put". A total of 45 atomic actions are supported by VirtualHome. Different actions take in different numbers of arg necessary for specifying an interaction. Associated with each arg is a unique id specifying the corresponding node in the environment graph, in case of multiple instances of the same object class are present in the graph. For the sake of simplicity, we omit the id in the remaining discussions of this paper and allow automatic assignment by the environment. An example program is shown in Appendix **??**.

Evaluated Tasks We use the knowledge base collected by VirtualHome for evaluation. The knowledge base contains household activities crowd-sourced from Amazon Mechanical Turk (MTurk). The MTurk workers were asked to provide natural language descriptions of daily household activities and all actionable steps necessary for completing the activities. The descriptions are both given as high-level task descriptions and step-by-step instructions. We omit the use of step-by-step instructions in this work as we desire direct extraction of executable programs from only task descriptions. For evaluations, we randomly sample a subset of 88 high-level tasks, each having one or more annotated ground-truth programs. The remaining 204 tasks are used as *demonstration set*, from which we are allowed to select as example(s) for prompting language models. Note that no training or fine-tuning is performed using these tasks and their annotations.

2.2 METRICS

A program that commands the agent to wander around in a household environment is highly executable but may not complete the desired task. On the other hand, a program composed of step instructions from knowledge bases can likely complete the task but cannot be executed. The reason is that instructions can be ambiguous and may lack necessary common-sense actions (i.e. fridge must be opened before agent can grab things from it). To this end, we consider two axes for evaluation: **executability** and **correctness**.

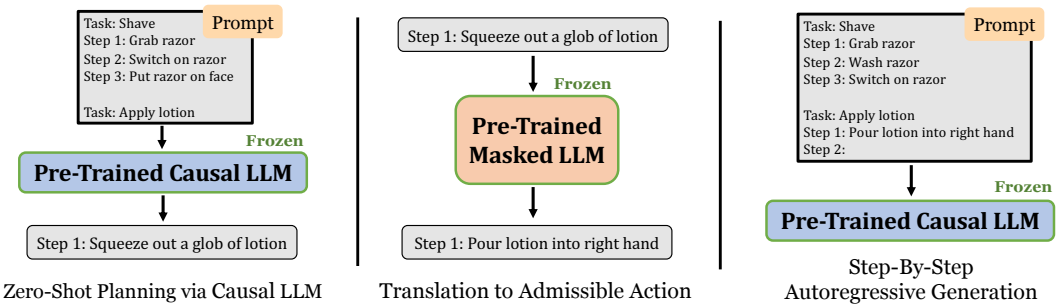
Executability Executability measures whether an action plan can be executed in the environment. A plan is not executable if 1) it is not syntactically correct, 2) it uses an action or object not recognized by environment, or 3) it violates environment constraints (i.e. fridge example above). We report the average executability across all 88 tasks and across all 7 VirtualHome scenes.

Correctness Unlike most embodied environments where the completion of a task can be easily judged, the ambiguous and multimodal nature of natural language task specification makes it impractical to obtain a gold-standard measurement of correctness. One approach could be measuring similarity of the final environment state produced by executing predicted and ground-truth programs, but VirtualHome initializes an environment differently based on the to-be-executed program, making comparisons difficult if measured in such way. Therefore, we conduct human evaluation for the highlighted methods. For the remaining methods and ablation studies, we rely on a match-based metric that measures how similar a generated program is to human annotations. Specifically, we follow Puig et al. (2018) and calculate the longest common subsequence (LCS) between two programs, normalized by the maximum length of the two. In the presence of multiple ground-truth programs for a single task, we take the maximum LCS across the ground-truth programs. However, we note that the majority of the tasks only have one ground-truth annotation, but there are often many plausible ways to complete a certain task, making this metric imperfect at evaluation program correctness[2]. Although correlation between the two is shown by Puig et al. (2018), we consider it only as a proxy metric in replacement of unscalable human evaluation.

3 METHOD

In this section, we investigate the possibility of extracting actionable knowledge from pre-trained language models without further training. We first give an overview of the common approach to

[2]Although LCS has a mathematical range of $[0, 1]$, we measure the LCS between different ground-truth programs for the same task and find an empirical maximum of 0.489.



Zero-Shot Planning via Causal LLM | Translation to Admissible Action | Step-By-Step Autoregressive Generation

Figure 2: We investigate the possibility of extracting actionable knowledge from pre-trained language models without any additional training. We first show surprising finding that large language models (LLMs) can decompose high-level tasks into sensible low-level action plans (left). To make the action plans executable, we propose to translate each step into admissible action via another LLM (middle). The translated action is appended to the original prompt used for generating the remaining steps (right).

query large language models (LLMs) and how it may be used for embodied agents. Then we describe an inference-time procedure that addresses several deficiencies of the LLM baseline and offers better executability in embodied environments. We break down the proposed procedure into three individual components, each discussed in Sec 3.2, 3.3, and 3.4.

Since LMs excel at dealing with natural language text instead of the specific format required by VirtualHome as described in Section 2.1, we only expose natural language text to LMs. To do this, we define a mapping for each atomic action that parses a natural language phrase to the required format. For instance, "Walk to living room" is converted to "[Walk] ⟨living_room⟩(1)". When an LM output cannot be parsed to any of the allowed action, the entire program is considered syntactically incorrect and thus not executable.

3.1 Prompting

Previous works have shown that large language models pre-trained on a colossal amount of data contain useful world knowledge that can be probed to perform various down-stream tasks (Radford et al., 2019; Brown et al., 2020). Notably, autoregressive LLMs can even perform in-context learning, an approach to solve tasks using only contextual information without gradient updates (Brown et al., 2020). Contextual information is given as part of the input prompt and LMs are asked to complete the remaining text. It often consists of natural language instructions and/or a number of examples containing the desired input/output pairs.

We adopt the same approach to query LLMs to generate action plans for high-level tasks. Specifically, we prepend one example task description sentence and its annotated action plan from the *demonstration set* to the query task description, as shown in Fig 2. To obtain text completion results, we sample from autoregressive LLM using temperature sampling and nucleus sampling (Holtzman et al., 2019). We refer to this LM as **Planning LM** and the approach using this LM for plan generation as **Vanilla *[LM]***, where ***[LM]*** is replaced by specific language model such as GPT-3 or Codex.

To further improve the quality of the generated output, we follow Chen et al. (2021) that uses LMs for program synthesis to sample multiple output for each task. However, unlike prior works in program synthesis that choose the sample with highest unit test pass rate, we only consider the setting where one sample is allowed to be evaluated for each task. This is because repetitive trial-and-errors can be dangerous in the real world, and executing many action plans is equivalent to probing the environment for privileged information, which is often considered not viable.

3.2 Robust Parsing by Semantic Translation

One issue arises when naively following the above approach to generate action plans for high-level tasks: the action plan is often not executable because LMs are allowed to generate free-form text. Therefore, most of the time the output cannot be mapped to one unambiguous actionable step. And many reasons can cause such failures: 1) the output does not follow pre-defined mappings of any atomic action (i.e. "I first walk to the bedroom" does not follow "Walk to ⟨PLACE⟩"), 2) the output may refer to atomic action and objects using words unrecognizable by the environment (i.e. "Clean

the dirty dishes in the sink" where "clean" and "dirty dishes in the sink" cannot be mapped to precise action and object), 3) the output contains lexical ambiguous words (i.e. "Open TV" should instead be "Switch on TV"), or 4) the output may use disallowed action (i.e. "Microwave the cup").

Instead of developing a set of rules to transform the free-form text into admissible action steps, we propose to again leverage world knowledge learned by large language models to semantically translate the action. For each step in the action plan $\hat{a}$, we aim to find the most similar admissible environment action a_e as measured by cosine similarity:

$$\underset{a_e}{\operatorname{argmax}} \frac{f(\hat{a}) \cdot f(a_e)}{\|f(\hat{a})\| \|f(a_e)\|} \text{ where } f \text{ is an embedding function.}$$

To embed the output text and environment actions, we use a BERT-style LM (Devlin et al., 2018; Liu et al., 2019) trained with Sentence-BERT (Reimers & Gurevych, 2019) objective because of its suitability for sentence modeling. The sentence embedding is obtained by mean-pooling the last layer hidden states across all tokens. We refer to this LM as **Translation LM**. Note that this is a different LM than the GPT-style Planning LM discussed in the text so far. Using a single LM for both purposes could as well be possible and likely more efficient, but we leave such investigation to future works. While the set of actions in our environment is discrete and possible to exhaustively enumerate, sampling or projection can be employed in larger discrete or continuous action spaces.

Since Translation LM can guarantee the parsed action is allowed by the environment, we can trade-off semantic soundness of an LM step by how likely it can be mapped to an admissible action in the environment. This can be achieved by a simple modification to the scheme that we use to choose the *best* sample from the LM output. Instead of only using mean token log probability as a ranking metric, we choose the sample with the highest score calculted as $s = C + \beta \cdot logprob$, where C is the cosine similarity to the closest allowed action and β is a weighting parameter.

3.3 Autoregressive Trajectory Correction

Translating each step of the program after the entire program has been synthesized is analogous to open-loop planning and is subject to compounding errors. In practice, LLMs might output compounded instructions for a single step, even though it cannot be completed using one admissible action in the environment. To this end, we can instead interleave *plan generation* and *action translation* to allow for automatic trajectory correction. At each step, we first query Planning LM to generate k samples for a single action. Then we calculate score s for each sample using Translation LM and append the translated action to the unfinished text completion. This way all subsequent steps will be conditioned on admissible actions instead of free-form text output generated by Planning LM. Furthermore, we can use Translation LM to detect out-of-distribution actions, those outside the capabilities of a robot, and terminate a program early instead of mapping to a faulty action. This can be easily implemented by setting a threshold ϵ such that if $C^t_{\text{max}} < \epsilon$ at step t, the program is terminated early. We empirically show this leads to better executability while maintaining similar correctness of the generated action plans.

3.4 Dynamic Example Selection for Improved Knowledge Extraction

So far in the text, we always give the same example in the prompt for all evaluated high-level tasks. However, consider the task of "ordering pizza". Prompting LLMs with this task may give the assumption that the agent is initialized in front of a computer, and the LLMs may guide the agent to search for a pizza store and click "checkout my cart". Although these are reasonable and feasible in the real world, such assumption cannot always be made as these interactions may not be supported in simulated environments like VirtualHome. In fact, the closest series of actions that human experts give may be "walking to a computer", "switching on the computer", and "typing the keyboard". Without being finetuned on these data, LLMs would often fail at these tasks. To provide weak supervision at inference time, we propose to use Translation LM to select the most similar task from the *demonstration set* to be used as the example in the prompt. Specifically, we choose the task whose high-level description matches closely the query task as measured by cosine similarity. This allows Planning LM to reason how to perform a similar task given a human-annotated example. In our evaluations, we make sure the demonstration set is not overlapping with our test queries.

Figure 3: Visualization of VirtualHome programs generated by our approach. The top row shows the execution of the task "Complete Amazon Turk Surveys", and the bottom row shows the task "Get Glass of Milk". We show LLMs not only can generate sensible action plans given only high-level tasks but also contains the actionable knowledge that can be extracted for grounding in embodied environments.

Language Model	Executability	LCS	Correctness[4]
Vanilla GPT-2 117M	18.66%	3.19%	–
Vanilla GPT-2 1.5B	39.40%	7.78%	19.51%
Vanilla GPT-Neo 2.7B	29.92%	11.52%	–
Vanilla GPT-3 12B	25.87%	13.40%	30.98%
Vanilla GPT-3 175B	7.79%	17.82%	**65.19%**
Vanilla Codex 2.5B	17.62%	15.57%	–
Vanilla Codex 12B	18.07%	16.97%	56.06%
Annotated GT	100.00%	N/A	54.80%
Translated GPT-3 175B	73.05%	24.09%	–
Translated Codex 12B	**78.57%**	**24.72%**	34.97%

Table 1: Human-evaluated correctness and evaluation results in VirtualHome. Although action plans generated by GPT-3 and Codex can even surpass the annotated GT in correctness measure, they are rarely executable. By translating the naive action plans, we show an important step towards grounding LLMs in embodied environments, but we observe room to achieve this without trading executability for correctness. We also observe a failure mode among smaller models that lead to high executability.

Combining the various improvement discussed above, we refer to the final approach as **Translated *[LM]***, where ***[LM]*** is replaced by specific language model used such as GPT-3 and Codex.

4 Results

In this section, we first show that language models can generate sensible action plans for many high-level tasks, even without any additional training. Then we highlight its inadequacy when naively applied to embodied environments and demonstrate how this can be improved by again leveraging world knowledge learned by LLMs. Visualization of generated programs are shown in Fig 3.

Sampling from LMs Pre-trained LMs are sensitive to sampling parameters and the specific example given in the prompt. For all evaluated methods, we perform hyper-parameter search over various sampling parameters, and for methods using fixed prompt example, we report metrics averaged across three randomly chosen examples. To select the best run for each method, we rank the runs by $LCS + executability$, each normalized by human-expert scores[3]. Further details can be found in Appendix **??**.

Model Choices We find empirically the combination of Codex-12B and Sentence-RoBERTa-355M work well in our setting. Codex is accessed using OpenAI API. The remaining models are accessed through open-source packages, Hugging Face Transformers (Wolf et al., 2019) and SentenceTransformers (Reimers & Gurevych, 2019), without additional modifications.

[3]See footnote 2.

[4]Due to the high cost of human evaluations, we only evaluate a subset of models.

4.1 DO LLMS CONTAIN ACTIONABLE KNOWLEDGE FOR HIGH-LEVEL TASKS?

We first investigate whether LLMs can generate sensible action plans expressed in free-form language. We use the approach described in Section 3.1 to query pre-trained LLMs. To evaluate the correctness of generated action plans, we conduct human evaluations. For each model, we ask 10 human annotators to determine – by answering "Yes" or "No" – whether each task can be completed using provided action steps. To provide a reference of how humans might rate the action plans provided by other humans, we also ask annotators to rate the ground-truth action plans provided in the VirtualHome dataset for the same set of tasks. In contrast to the free-form text output by LLMs, the ground-truth action plans from VirtualHome are generated via a graphical programming interface that enforces strict syntax, although annotators were allowed to compose necessary actions.

We show the human evaluation results in Fig 1, where y-axis shows correctness averaged across all tasks and all annotators. Surprisingly, when LLMs are large enough and without imposed syntactic constraints, they can generate highly realistic action plans whose correctness – as deemed by human annotators – even surpasses human-labeled ground-truth. Yet another interesting finding is that Codex outperforms GPT-3 significantly under the same number of model parameters. One hypothesis could be that by fine-tuning on structured data (docstrings and code), Codex specializes at decomposing a high-level objective into a number of basic operations, even with those operations described in natural language. We also observe some level of correctness for smaller models such as GPT-2. However, inspection of its produced output indicates that it often generates significantly shorter plans by ignoring common-sense actions or by simply rephrasing the given task (e.g. the task "Go to sleep" produces only a single step "Go to bed"). These failure modes sometimes mislead human annotators to mark them correct as the annotators may ignore common-sense actions in their judgment as well, resulting in a higher correctness rate than the quality of the output shows.

4.2 HOW EXECUTABLE ARE THE LLM ACTION PLANS?

We analyze the executability of LLM plans by evaluating them in all 7 household scenes in VirtualHome. As shown in Table 1, we find action plans generated naively by LLMs are generally not very executable. Although smaller models seem to have higher executability, we find that the majority of these executable plans are produced by ignoring the queried task and repeating the given GT example of a different task. This is validated by the fact that smaller models have lower LCS than larger models despite having higher executability, showing that this failure mode is prevalent among smaller models. In contrast, larger models do not suffer severely from this failure mode. Yet as a result of being more expressive, their generated programs are substantially less executable.

4.3 CAN LLM ACTION PLANS BE MADE EXECUTABLE BY ACTION TRANSLATION?

In this section, we evaluate the effectiveness of our proposed procedure of action translation. We first create a bank of all allowed 47522 action steps in the environment, including all possible combinations of atomic actions and allowed arguments/objects. Then we use an off-the-shelf Sentence-RoBERTa (Liu et al., 2019; Reimers & Gurevych, 2019) as Translation LM to create embeddings for actions and output text. For better computational efficiency, we pre-compute the embeddings for all allowed actions, leaving minor computation overhead for our procedure over the baseline methods at inference time. As shown in Table 1, executability of generated programs is significantly improved. Furthermore, we also observe improved LCS because the translated action steps precisely follow the program syntax and thus are more similar to ground-truth annotations. One sample output is shown in Fig 1 and a larger random subset of generated samples can be found in Appendix **??**.

To validate their correctness, we again perform human studies using the same procedure in Sec 4.1. Results are shown in Table 1. We find that despite more similar to GT, the programs are deemed less correct by humans. By examining the generated output, we observe two main sources of errors. First, we find Translation LM is poor at mapping compounded instructions to a succinct admissible action. This is partly due to that Translation LM is trained on a much smaller dataset and contains much a smaller number of parameters, so we expect further improvement by using a larger pre-trained model for translation. The second source of error comes from imperfect expressivity of the environment; we find that for a many tasks we evaluate, certain necessary actions or objects are not implemented. This is also reflected by out human evaluation results of the GT programs, as only half of the programs are considered complete by our human annotators.

5 ANALYSIS AND DISCUSSIONS

5.1 ABLATION OF DESIGN DECISIONS

We perform ablation studies to show the effectiveness and necessity of three components of our proposed procedure, each described Sec 3.2, 3.3, and 3.4. As shown in Table 2, leaving out any of the three components would all lead to decreased performance in both executability and LCS. Notably, not doing action translation leads to the most significant executability drop, showing the importance of action translation in extracting executable action plans from LLMs.

Methods	Executability	LCS
Translated Codex 12B	**78.57%**	**24.72%**
- **w/o** Action Translation	31.49%	22.53%
- **w/o** Dynamic Example	50.86%	22.84%
- **w/o** Iterative	55.19%	24.43%

Table 2: Ablation of three proposed techniques.

5.2 ANALYSIS OF PROGRAM LENGTH

Shorter programs have a natural advantage of being more executable. Consider a task "wash hands" and a corresponding program that only commands an agent to "go to bathroom" without additional steps. The program is obviously incorrect yet trivially executable. To validate our approach does not simply generate very short program, we calculate the average program length across the 88 evaluated tasks. Results are shown in Table 3. In addition to the failure mode discussed in Sec 4.2 that leads to incorrect yet executable programs, smaller LMs such as GPT-2 also generate programs significantly shorter than larger models, making them more executable. In contrast, larger models like Codex-12B generate more expressive program of high correctness, but they often suffer from executability. We show action translation can lead to benefits of both worlds, generating programs that are highly executable while maintaining similar expressiveness in terms of program length.

Methods	Executability	Average Length
Vanilla GPT-2 1.5B	39.40%	4.24
Vanilla Codex 12B	18.07%	7.22
Translated Codex 12B	78.57%	7.13
Ground-Truth	100.00%	9.66

Table 3: Average executability & program length of different methods and human annotations.

5.3 CAN LLMS GENERATE ACTIONABLE PROGRAMS BY FOLLOWING DETAILED INSTRUCTIONS?

Prior works often focus on translating step-by-step instructions into executable programs. We evaluate LLMs under this setting using a prompt format shown in Appendix **??**. Although this setting is easier as it does not require rich actionable knowledge, detailed instructions can help resolve much ambiguity of exactly how to perform a high-level task when multiple solutions are possible. Therefore, **Translated Codex 12B** achieves executability of 78.57% and LCS of 32.87%, where LCS sees a considerable bump from the setting without detailed instructions. Surprisingly, the LCS result is very close to that of a supervised LSTM (Hochreiter & Schmidhuber, 1997) baseline from VirtualHome trained on human-annotated data, which is at 34.00%. Note that since code to train the baseline and the specific train/test split is not publicly released, we only show results reported in Puig et al. (2018) as a reference. We also cannot compare executability as it is not reported.

5.4 IS ACTION TRANSLATION NECESSARY FOR ACTIONABLE KNOWLEDGE GROUNDING?

The investigations of this paper are two-fold: 1) Is actionable knowledge present in LLMs? 2) Can we ground this actionable knowledge in interactive environment? In this section, we focus our attention on the second question by conditioning on the assumption that first question is true. To do this, since successful execution of *correct* action plans directly measures grounding, we select only the *correct* plans generated by LLMs and measure how executable they are. We deem an action plan to be *correct* if 70% or more human annotators decide it is correct.

Methods	# of C	# of C and E	E / C
GPT-2 1.5B	8	0	0.00%
GPT-3 12B	24	2	8.33%
GPT-3 175B	**68**	5	7.35%
Codex 12B	45	8	17.78%
Translated Codex 12B	18	**15**	**83.33%**

Table 4: Count of correct/executable programs and percentage of executable among correct. C indicates correct and E indicates executable.

As shown by Table 4, when an LM is not large enough (e.g. GPT-2), not only it contains little actionable knowledge, but this knowledge cannot be grounded at all. GPT-3 and Codex, on the other hand, can generate highly correct action plans in free-form language. However, they do not have the capability to ground their actionable knowledge in interactive environments. What's more interesting, by comparing GPT-3 of both 12B parameters and 175B parameters, ratio of executable plans does not improve with the parameter count. This shows that simply training larger models does not necessarily lead to better knowledge grounding. In the meantime, action translation offers a promising way towards grounding actionable knowledge by producing highly executable plans. However, we again note that it comes at a trade-off of producing less correct plans as compared to its vanilla counterpart, and we hope to see future endeavors for bridging the gap.

6 Related Works

Large-scale natural language modeling has witnessed rapid advances since the inception of the Transformer architecture (Vaswani et al., 2017). It has been shown by recent works that large language models (LLMs) pre-trained on large unstructured text corpus not only can perform strongly on various down-stream NLP tasks (Devlin et al., 2018; Radford et al., 2019; Raffel et al., 2019; Brown et al., 2020) but also can internalize an implicit knowledge base containing rich information about the world (Roberts et al., 2020). The encoded world knowledge can be used to model relations of entities (Li et al., 2021) and retrieve matching visual features (Ilharco et al., 2020). Furthermore, it is even shown to be valuable priors when applied to diverse tasks from different modalities (Lu et al., 2021; Tsimpoukelli et al., 2021).

At the same time, there has also been growing interest and development in grounding language in embodied environment. A series of prior works have investigated the possibility of parsing language instructions into formal logic to resolve various linguistic ambiguities for embodied agents (Artzi & Zettlemoyer, 2013; Misra et al., 2015; Tenorth et al., 2010). However, they often scale poorly to complex tasks and environments. Recently, more research efforts have been put into creating better and more realistic environments with the goal to further advances in this area (Puig et al., 2018; Shridhar et al., 2020a;b; Kolve et al., 2017; Savva et al., 2019). At the same time, by leveraging the better representation power of neural architectures, a number of works have looked into creating instruction-following agents that can perform manipulation (Lynch & Sermanet, 2020), navigation (Majumdar et al., 2020), or both (Suglia et al., 2021; Hill et al., 2020).

Notably, most of these prior works do not leverage full-blown pre-trained LLMs (Suglia et al., 2021) or do not scale to complex human activities (Hill et al., 2020; Lynch & Sermanet, 2020). Perhaps more importantly, few works have evaluated LLMs in an embodiment setting that realizes the full potential of the world knowledge these models contain: the tasks evaluated are often "pick", "grab", "open", and etc, which do not resemble the highly diverse activities that humans perform in daily lives. The development of VirtualHome environment Puig et al. (2018) enables such possibility. However, relevant works (Puig et al., 2020; Liao et al., 2019) rely on human-annotated data and perform supervised training from scratch. Due to the lack of rich world knowledge, these models can only generate action plans given step-by-step instructions of how to act or video demonstrations. In this work, we take a step further by conditioning only on the high-level descriptions and by extracting executable action plans from LLMs without any additional training.

7 Limitations and Conclusion

There are several notable limitations of this work. First, although our approach presents a viable way to ground world knowledge in embodied environments, it is still a trade-off rather than one best solution since we observe considerable drop in correctness. Second, we focus on high-level to mid-level grounding, assuming there is a controller that can execute mid-level tasks (such as "grab cup"). Our work does not investigate usefulness of LLMs for low-level sensorimotor behavior grounding. The third limitation is that we do not incorporate observation context or feedback into our models. To some extent, we approach LLMs in the same way as how VirtualHome asks human annotators to give action plans for a given huamn activity by *imagination*, in which case the human-generated action plans also do not incorporate observation context. However, we do see incorporating observation context for complex activities as an exciting future direction.

REFERENCES

Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

BIG-bench collaboration. Beyond the imitation game: Measuring and extrapolating the capabilities of language models. *In preparation*, 2021. URL `https://github.com/google/BIG-bench/`.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Kohd, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Gabriel Ilharco, Rowan Zellers, Ali Farhadi, and Hannaneh Hajishirzi. Probing text models for common ground with visual representations. *arXiv e-prints*, pp. arXiv–2005, 2020.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Belinda Z Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

Yuan-Hong Liao, Xavier Puig, Marko Boben, Antonio Torralba, and Sanja Fidler. Synthesizing environment-aware activities via activity sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6291–6299, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *European Conference on Computer Vision*, pp. 259–274. Springer, 2020.

Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. Environment-driven lexicon induction for high-level instructions. 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)*, pp. 992–1002, 2015.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration. *arXiv preprint arXiv:2010.09890*, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347, 2019.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020a.

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

Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *arXiv preprint arXiv:2108.04927*, 2021.

Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *2010 ieee international conference on robotics and automation*, pp. 1486–1491. IEEE, 2010.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *arXiv preprint arXiv:2106.13884*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.