ISSN 1064-5624, Doklady Mathematics, 2022, Vol. 106, Suppl. 1, pp. S85–S90. © The Author(s), 2022. This article is an open access publication.
ISSN 1064-5624, Doklady Mathematics, 2022. © The Author(s), 2022. This article is an open access publication.
Russian Text © The Author(s), 2022, published in Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya, 2022, Vol. 508, pp. 94–98.

Application of Pretrained Large Language Models in Embodied Artificial Intelligence

A. K. Kovalev[a] **and A. I. Panov**[b,*]

Presented by Academician of the RAS A.L. Semenov

Received October 28, 2022; revised October 31, 2022; accepted November 3, 2022

Abstract—A feature of tasks in embodied artificial intelligence is that a query to an intelligent agent is formulated in natural language. As a result, natural language processing methods have to be used to transform the query into a format convenient for generating an appropriate action plan. There are two basic approaches to the solution of this problem. One is based on specialized models trained with particular instances of instructions translated into agent-executable format. The other approach relies on the ability of large language models trained with a large amount of unlabeled data to store common sense knowledge. As a result, such models can be used to generate an agent's action plan in natural language without preliminary learning. This paper provides a detailed review of models based on the second approach as applied to embodied artificial intelligence tasks.

Keywords: embodied artificial intelligence, large language models, common sense knowledge, construction of action plans

DOI: 10.1134/S1064562422060138

1. INTRODUCTION

In recent years, increased interest has been shown in embodied artificial intelligence (EAI) tasks, which are generally represent the operation of objects in human-oriented environments (household tasks) or the displacement of objects and navigation in homes or open areas. A distinctive feature of EAI tasks is that instructions describing a task to be completed or a goal to be reached that are transferred to an embodied intelligent agent (EIA) are formulated in natural language. Accordingly, natural language processing techniques have to be used to transform the instructions into a form convenient for EAI. There are two approaches to the solution of this problem.

One relies on specialized models trained for generating a plan of agent's actions based on instructions. Examples are techniques for using templates of possible actions and determining arguments of these actions [1] or models generating token sequences (Seq2seq) [2].

The other approach is based on the fact that modern large language models (LLMs) [3, 4] pretrained with large corpora of unlabeled texts produce good results on tasks for which they were not initially designed after few-shot learning [5] or without learning at all [6]. This is achieved due to the ability of such models to store common sense knowledge. In modern works, this property of LLMs is used in EAI tasks, for example, in [7].

This paper provides a detailed review of models based on the second approach to EAI tasks in which pretrained LLMs are used to generate a plan of EIA actions in some environment.

2. APPROACHES OF BASED ON PRETRAINED LLM

In Zero-Shot Planners [7], it is proposed using LLMs for grounding high-level tasks expressed in natural language to a set of elementary actions executable by an intelligent agent. As a result, given a task description, LLM must to construct an action plan for implementing the task. Here, the LLM is not additionally trained. As an environment where the intelligent agent acts, we use the Virtualhome simulator [8]. A plan is constructed iteratively. First, a specially formulated query with a task description made in natural language is fed as input to the model. Then the model generates in free form a description of the action to be executed at the first step. For the resulting action description, a special vector representation is calculated [9], for which an action with the closest vector representation

[a] *Artificial Intelligence Research Institute, Moscow, Russia*

[b] *Federal Research Center "Computer Science and Control," Russian Academy of Sciences, Moscow, Russia*

**e-mail: panov@airi.net*

is sought in the set of elementary actions. Next, the description of the obtained action is added to the query text and the procedure is repeated. A query to the LLM is formulated in the form of a hint with a task example and an action plan for its execution placed at the beginning of the query and with the description of the current task added to the end of the query. Primary attention is given plan generation. The agent is assumed to be able to perform elementary actions.

In G-PlanET [10], LLMs are also used for action plan generation. However, in contrast to [7], emphasis is placed on binding to a particular environment, rather than to only agent's actions. A modification of an ALFRED task [11] is used in which the scene is represented as a table with an attached list of all scene objects with their type, position, orientation, and parent object (on which the given object lies/in which it is placed). At the planning stage, the scene table is unfolded in lines and is added to the description of the task. The resulting text in the form of a query is fed to the input of the LLM. Thus, a query in G-PlanET [10] consists of a generated tabulated representation of the scene with an operating intelligent agent in it and a task description in natural language. A plan is generated iteratively. The result of the current step is concatenated with the query for this step and is fed to the input of the model for generating the next step. A similar approach is used in EA-APG (environmentally-aware action plan generation) [12], but the scene description consists of only a list of objects.

In the SayCan architecture [13], a key stage is training an agent to execute elementary actions. An action consists of three parts: a strategy, i.e., a sequence of instructions for the displacement of an intelligent agent or its movable parts; a natural language description; and an affordance function returning the probability of successful execution of the action in the current state of the environment. The generation of a plan is similar to the process implemented in Zero-Shot Planners [7], with the only difference being that the LLM does not generate a description of the next action, but rather estimates the probability that an elementary action is useful for executing the task in question. Such an estimate is produced for all possible elementary actions and is multiplied by the successful action execution probability obtained from the utility function correspondences. The next action is chosen to be the one with the maximum estimate value. A distinctive feature of this work is that experiments were conducted in both virtual environment and on robotic platforms in a realistic environment. It should also be noted that a query consists not of a single example as in Zero-Shot Planners [7], but rather contains several examples. Another difference in the formation of a query is that it represents not a task description with an action plan, but rather a dialogue between the user and the intelligent agent in which the former asks a question, for example, "How can you bring some snack to me?", and the agent lists the actions necessary for executing this task. Additionally, the authors use the approach to query formation proposed in [14]. According to this approach, a task execution description is added to the query in addition to the task and its solution. The use of such a hint in SayCan [13] improves the performance of the model for tasks involving negation or reasoning. A limitation of this approach is that, as was shown in [14], improvements are demonstrated only for LLMs with more than 100 billions of parameters.

In the ProgPromt architecture [15], the basic idea is that a query in represented in the form of a Python-like code. A query consists of a description of available actions in the form of imported corresponding software modules, a list of objects in the scene, task examples, and their execution in the form of software modules. This form of query is justified by the fact that LLMs, such as GPT-3 with 175 billion parameters [5], are trained on large amounts of data from open software code repositories. A similar approach is used in the CaP model [16], which generates an agent's strategy. A distinctive feature is that the resulting programs are executable software codes, and they can be executed in a hierarchical manner.

Some works, for example, the Socratic Model [17] and Inner Monologue [18], propose not a particular model, but rather an approach to the construction and union of a set of models. For example, pretrained models involving various modalities (sound, text, image, etc.) are used in the Socratic Model [17]. Such models can be combined in systems capable of solving tasks going beyond the scope of tasks for each individual model. This is achieved by developing an interface for data exchange between the models. As an example of robotic task, planning displacements of objects over a table is considered. The Pybullet simulator [19] is used to detect objects in a scene. Given an image, its description is produced by applying the ViLD approach [20], after which the scene description in the form of a query is fed to LLM for generating an action plan by analogy with Zero-Shot Planners [7] and SayCan. Next, the plan is executed following a CLIPort-like strategy [21, 22]. A query consists of a scene description in the form of a python-like list of objects, examples of tasks formulated in natural language, and their implementation in the form of a pseudocode.

The Inner Monologue architecture [18] makes use of text-like feedback obtained from either the environment (scene description, success of action execution) or from the user (refinement of agent's actions). The general approach remains fixed, while, depending on the considered task, different models are used for implementation. The basic idea is that feedback from the environment in the form of text is added iteratively to the input query used for planning LLM. A query with a scene description and task examples is used in manipulations with objects (in an environment with a virtual or actual table). For executing a task in a realis-

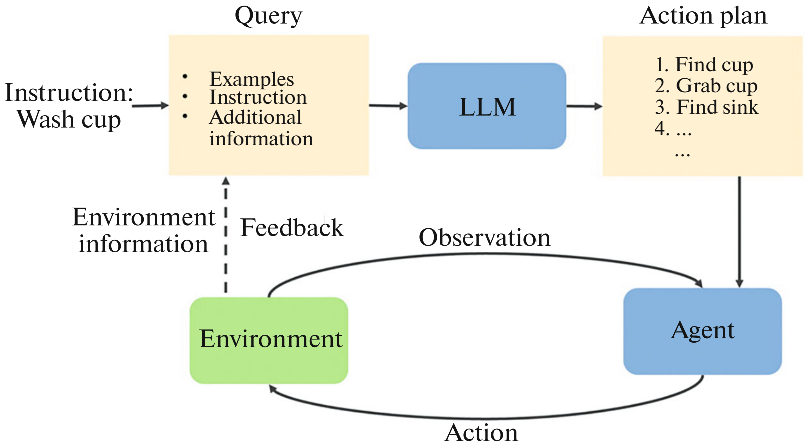

Fig. 1. General architecture of using large language models (LLM) in the task of generating an embodied agent's action plan.

tic kitchen, a query is formulated in the form of a dialogue between the user and the embodied agent.

The LM-Nav [23] relies on an approach that falls into Socratic Models [17], in which case separately pretrained models are joined in a single system to be used for text and image navigation. LLM GPT-3 [5] is employed for generating a sequence of text labels based on natural language instructions, a visual language model assigns text labels to images obtained by the agent [24], and an image navigation model [25] generates and executes a plan of the agent's displacement. Three examples of extracted text labels are used as a query for LLM.

3. CLASSIFICATION OF LLM FOR PLANNING TASK

A classification of the considered approaches by types of queries, testing environments, and used LLMs is presented in Table 1. Generalizing the data given in Table 1, we should note that, in the general form, a query to a language model can consist of the following parts:

1. a scene description, which is simply a list of available objects possibly supplemented with their properties;
2. a list of actions available to the embodied agent;
3. examples of tasks posed for the embodied agent;
4. examples of executing posed tasks.

The query itself can be expressed as a plain text or a dialogue. In addition to natural language queries, it is possible to use queries representing a pseudo-code or an executable software code written, for example, in the Python language.

It should be noted that choosing a proper query is a rather difficult task, and its result can be affected by seemingly minor modifications, such as the numbering of the list of planned actions and additions of line break symbols (see the examples in SayCan [13]).

All the considered approaches can be described by a common architecture with LLM for the task of planning the embodied agent's actions represented in Fig. 1.

At the first stage, a natural language instruction describing the task for the embodied agent is used to form a query for LLM. In the simplest case, a query consists of task examples with execution plans in terms of instructions available to the embodied agent [7, 13, 23]. A query of more complex structure can include additional information on the environment, for example, a list of present objects and their properties [10, 12, 15–18] or available actions of the agent [15, 16]. Next, the query is fed to LLM, which iteratively generates an action plan. It should be noted that queries are mainly formulated in natural language [7, 10, 12, 13, 17, 18, 23], but there are approaches using pseudo-codes [16, 17] or software codes [15] for this purpose. At the second stage, the agent implements the generated plan. This formulation means that the action plan is completely generated prior to its implementation in the environment and is not modified in the course of the implementation. This may lead to a situation when the agent gets stuck at some stage of the plan implementation, which, in turn, may lead to the initial task failed to be executed. A possible way out of this problem is to use feedback from the environment (dashed arrow in Fig. 1) at every iteration generating the next agent's action. As feedback, it is possible to use information on the possibility of executing a particular action [13] or a report concerning the correct execution of an action or the variation in the object state [18].

CONCLUSIONS

The considered approaches to the use of LLM for generating action plans yield moderately good results in both virtual environments and implementations on robotic platforms. Nevertheless, a quantitative comparison of these works is complicated by the fact that they involve different environments (except for several works). Note that there are well-known benchmark

Table 1. Comparisons of algorithms for embodied artificial intelligence tasks based on pretrained LLMs

Algorithm	Language model	Environment	Robot implementation	Type of query	Navigation	Interact. with environment
Zero-Shot Planners [7]	GPT-3 175B [5] Codex 12B [29]	VirtualHome [8]	–	Examples of tasks and their solutions	+	+
G-PlanET [10]	TaPEX [30]	ALFRED [11] (modification)	–	Scene description, task description	–	–
EA-APG [12]	GPT-3 [5]	VirtualHome [8]	–	Scene description, examples of tasks and their solutions	+	+
SayCan [13]	PALM 540B [4]	Realistic environment (kitchen)	Everyday Robots	Examples of tasks and their solutions formulated in the form of dialogue	+	+
ProgPromt [15]	GPT-3 175B [5]	VirtualHome [8]	–	Python-like code with description of admissible actions, scene, examples of tasks, and their implementation	+	+
Socratic [17]	GPT-3 175B [5]	Pybullet [19]	–	Scene description, examples of tasks, and their implementation on pseudo-code	–	+
Inner Monologue [18]	InstructGPT [31]	Pybullet [19]	–	Scene description, examples of tasks, and their implementation	–	+
	InstructGPT [31]	Realistic environment (table)	Everyday Robots	Scene description, examples of tasks, and their implementation	–	+
	PALM 540B [4]	Realistic environment (kitchen)	Everyday Robots	examples of tasks, and their implementation in dialogue form	+	+
CaP [16]	Codex [29] code-davinci-002	Realistic environment (drawing on white board)	UR5e	Python-like code with description of admissible actions, scene, examples of tasks, and their implementation	+	–
	Codex [29] code-davinci-002	Realistic environment (table)	UR5e	Python-like code with description of admissible actions, scene, examples of tasks, and their implementation	+	–
	Codex [29] code-davinci-002	Realistic environment (kitchen)	Everyday Robots	Python-like code with description of admissible actions, scene, examples of tasks, and their implementation	+	+
LM-Nav [23]	GPT-3 [5]	Realistic environment (street)	Clearpath Jackal UGV	Examples of tasks and their implementation	+	–

tasks with established quality metrics and comparison tables for testing embodied intelligent agents. Examples are ALFRED [11] and TEACh [26] for following natural language instructions, RoomR [27] and Benchbot [28] for object rearrangements, and others. Unfortunately, most of the indicated works are not presented in these benchmarks, which is largely explained by the complexity of organizing queries in various environments with a large number of objects and actions.

A promising future direction of research is as follows. First, the feedback should be made more complicated and trained in order to determine the quality of LLM responses. Second, the training of LLMs should be modified, namely, a regularizer modeling the reality of generated responses should be added to the loss function of the entire language model.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

OPEN ACCESS

REFERENCES

1. S. Y. Min et al., "Film: Following instructions in language with modular methods," arXiv preprint arXiv:2110.07342 (2021).
2. H. Liu et al., "LEBP—language expectation & binding policy: A two-stream framework for embodied vision-and-language interaction task learning agents," arXiv preprint arXiv:2203.04637 (2022).
3. J. Devlin et al., "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805 (2018).
4. A. Chowdhery et al., "Palm: Scaling language modeling with pathways," arXiv preprint arXiv:2204.02311 (2022).
5. T. Brown et al., "Language models are few-shot learners," Advances in neural information processing systems," **33**, 1877–1901 (2020).
6. J. Wei et al., "Finetuned language models are zero-shot learners," arXiv preprint arXiv:2109.01652 (2021).
7. W. Huang et al., "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," arXiv preprint arXiv:2201.07207 (2022).
8. X. Puig et al., "Virtualhome: Simulating household activities via programs," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 8494–8502.
9. N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," arXiv preprint arXiv:1908.10084 (2019).
10. B. Y. Lin et al., "On grounded planning for embodied tasks with language models," arXiv preprint arXiv:2209.00465 (2022).
11. M. Shridhar et al., "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 10740–10749.
12. M. Gramopadhye and D. Szafir, "Generating executable action plans with environmentally-aware language models," arXiv preprint arXiv:2210.04964 (2022).
13. M. Ahn et al., "Do as I can, not as I say: Grounding language in robotic affordances," arXiv preprint arXiv:2204.01691 (2022).
14. J. Wei et al., "Chain of thought prompting elicits reasoning in large language models," arXiv preprint arXiv:2201.11903 (2022).
15. I. Singh et al., "ProgPrompt: Generating situated robot task plans using large language models," arXiv preprint arXiv:2209.11302 (2022).
16. J. Liang et al., "Code as policies: Language model programs for embodied control," arXiv preprint arXiv:2209.07753 (2022).
17. A. Zeng et al., "Socratic models: Composing zero-shot multimodal reasoning with language," arXiv preprint arXiv:2204.00598 (2022).
18. W. Huang et al., "Inner monologue: Embodied reasoning through planning with language models," arXiv preprint arXiv:2207.05608 (2022).
19. E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics, and machine learning," GitHub Repository (2016).
20. X. Gu et al., "Open-vocabulary object detection via vision and language knowledge distillation," arXiv preprint arXiv:2104.13921 (2021).
21. M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," *Conference on Robot Learning* (PMLR, 2022), pp. 894–906.
22. A. Zeng et al., "Transporter networks: Rearranging the visual world for robotic manipulation," arXiv preprint arXiv:2010.14406 (2020).
23. D. Shah et al., "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," arXiv preprint arXiv:2207.04429 (2022).
24. A. Radford et al., "Learning transferable visual models from natural language supervision," *International Conference on Machine Learning* (PMLR, 2021), pp. 8748–8763.
25. D. Shah et al., "Ving: Learning open-world navigation with visual goals," *2021 IEEE International Conference on Robotics and Automation* (*ICRA*) (IEEE, 2021), pp. 13215–13222.

26. A. Padmakumar et al., “Teach: Task-driven embodied agents that chat,” Proceedings of the AAAI Conference on Artificial Intelligence (2022), Vol. 36, No. 2, pp. 2017–2025.
27. L. Weihs et al., “Visual room rearrangement,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 5922–5931.
28. B. Talbot et al., “Benchbot: Evaluating robotics research in photorealistic 3d simulation and on real robots,” arXiv preprint arXiv:2008.00635 (2020).
29. M. Chen et al., “Evaluating large language models trained on code,” arXiv preprint arXiv:2107.03374 (2021).
30. Q. Liu et al., “Tapex: Table pre-training via learning a neural SQL executor,” arXiv preprint arXiv:2107.07653 (2021).
31. L. Ouyang et al., “Training language models to follow instructions with human feedback,” arXiv preprint arXiv:2203.02155 (2022).

Translated by I. Ruzanova