ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2022, том 508, с. 94–99

ПЕРЕДОВЫЕ ИССЛЕДОВАНИЯ В ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ

УДК 004.8

ПРИМЕНЕНИЕ ПРЕДОБУЧЕННЫХ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ В ЗАДАЧАХ ВОПЛОЩЕННОГО ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

© 2022 г. А. К. Ковалёв[1], А. И. Панов[2,*]

Представлено академиком РАН А.А. Шананиным
Поступило 28.10.2022 г.
После доработки 31.10.2022 г.
Принято к публикации 03.11.2022 г.

Особенностью задач воплощенного искусственного интеллекта является формирование запроса к интеллектуальному агенту на естественном языке. Это приводит к необходимости использования методов обработки естественного языка для перевода этого запроса в формат, удобный для составления корректного плана поведения. Существует два основных подхода к решению этой задачи. Первый подход заключается в использовании специализированных моделей, обученных на конкретных примерах перевода инструкций в исполнимый агентом формат. Второй подход использует способность больших языковых моделей, обученных на большом объеме неразмеченных данных, хранить знания общего назначения (common sense). Это позволяет использовать такие модели для построения плана поведения агента по запросу на естественном языке без предварительного дообучения. В данной обзорной статье подробно рассматриваются модели, использующие второй подход в задачах воплощенного искусственного интеллекта.

Ключевые слова: воплощенные искусственный интеллект, большие языковые модели, знания общего назначения, построение плана поведения

DOI: 10.31857/S268695432207013X

1. ВВЕДЕНИЕ

В последние годы возрос интерес к задачам воплощенного искусственного интеллекта (ВИИ/embodied artificial intelligence), которые, в основном, представлены либо задачами оперирования объектами в человеко-ориентированных средах (household tasks), либо перемещением объектов и навигацией в помещениях или на открытой местности. Отличительной чертой задач ВИИ является формулирование инструкций, описывающих выполняемую задачу или достигаемую цель и передаваемых воплощенному интеллектуальному агенту (ВИА), на естественном языке. Такая постановка приводит к необходимости использовать техники обработки естественного языка для перевода инструкций в вид, удобный для использования ВИИ. Существуют два подхода к решению этой задачи.

Первый подход использует специализированные модели, обученные для получения плана действия агента на основе инструкции. Примерами могут служить техника использования шаблонов возможных действий и определения аргументов этих действий [1] или модели генерации последовательности токенов (Seq2seq) [2].

Второй подход основан на том, что современные большие языковые модели (БЯМ) [3, 4], предобученные на больших корпусах неразмеченных текстов, демонстрируют хорошие результаты на задачах, для решения которых они изначально не были спроектированы, после небольшого дообучения (few-shot learning) [5] или совсем без дообучения [6]. Это достигается за счет того, что такие модели хранят знания общего назначения (common sense). Современные работы используют это свойство БЯМ в задачах воплощенного искусственного интеллекта, например [7].

В данной статье мы подробно рассматриваем модели, относящиеся ко второму подходу к задачам ВИИ, которые используют предобученные большие языковые модели для генерации плана поведения ВИА в среде.

2. ПОДХОДЫ НА ОСНОВЕ ПРЕДОБУЧЕННЫХ БЯМ

В работе Zero-Shot Planners [7] предлагается использовать БЯМ для “заземления” высокоуровневой задачи, выраженной на естественном языке, на множество элементарных действий, доступных интеллектуальному агенту. В результате

[1] *Институт искусственного интеллекта AIRI, Москва, Россия*
[2] *Федеральный исследовательский центр “Информатика и управление” Российской академии наук, Москва, Россия*
* *E-mail: panov@airi.net*

по описанию задачи БЯМ должна построить план действий, приводящий к выполнению поставленной задачи, при этом подразумевается, что БЯМ не дообучается. В качестве среды, в которой действует интеллектуальный агент, используется симулятор Virtualhome [8]. Построение плана происходит итеративно: сначала на вход модели подается специально сформулированный запрос с описанием задачи на естественном языке, после чего модель генерирует в свободной форме описание действия, которое необходимо выполнить на первом шаге. Для полученного описания действия считается специальное векторное представление [9], для которого ищется действие из множества элементарных действий с наиболее близким векторным представлением. После этого описание полученного действия добавляется к тексту запроса, и процедура повторяется. Запрос БЯМ формулируется в виде подсказки, где в начале запроса идут пример задачи и план действий по ее выполнению, а в конце добавляется описание текущей задачи. Основное внимание авторы уделяют составлению плана, при этом предполагается, что агент уже умеет выполнять элементарные действия.

В G-PlanET [10] также рассматривается использование БЯМ для генерации плана действий, однако, в отличие от Zero-Shot Planners [7], акцент делается на привязке к конкретной среде, а не только к действиям агента. Используется модификация задачи ALFRED [11], в которой сцена представляется как таблица с перечислением всех доступных на сцене объектов с их типом, положением, ориентацией и родительским объектом (на чем лежит/ в чем находится данный объект). На этапе планирования таблица сцены построчно разворачивается и объединяется с описанием задачи. Полученный текст в виде запроса подается на вход БЯМ. Таким образом, в качестве запроса в G-PlanET [10] используется сгенерированное табличное представление сцены, в которой функционирует интеллектуальный агент, и описание задачи на естественном языке. План генерируется итеративно, результат текущего шага конкатенируется с запросом для этого шага и подается на вход модели для генерации следующего шага. Похожий подход используется в подходе EA-APG (environmentally-aware action plan generation) [12], однако в нем описание сцены состоит только из перечисления объектов.

В архитектуре SayCan [13] процесс отдельного обучения агента выполнению элементарных действий является одним из ключевых этапов. Действие состоит из трех частей: стратегии, т.е. последовательности команд по перемещению интеллектуального агента или его подвижных частей; описания на естественном языке и функции соответствия цели (affordance function), возвращающей вероятность успешного выполнения действия в текущем состоянии среды. Процесс генерации плана похож на процесс, реализованный в Zero-Shot Planners [7], с тем различием, что БЯМ не генерирует описание следующего действия, а дает оценку вероятности того, что элементарное действие полезно для выполнения поставленной задачи. Такая оценка выдается для всех возможных элементарных действий и умножается на вероятность успешного выполнения действия, полученную по функции соответствия (полезности). В качестве следующего действия выбирается действие, с максимальным значением оценки. Отличительной особенностью этой работы является то, что эксперименты проводились как в виртуальной среде, так и на робототехнических платформах в реальной среде. Так же стоит отметить, что запрос состоит не из одного примера как в Zero-Shot Planners [7], а содержит несколько примеров. Еще одним отличием в формировании запроса является то, что он формируется не как описание задачи с планом действий, а как диалог между пользователем и интеллектуальным агентом, в котором пользователь задает вопрос, например, “Как бы ты мог принести мне перекус?”, а агент перечисляет действия, необходимые для решения поставленной задачи. Также авторы применили подход к формированию запроса, предложенный в [14]. Данный подход заключается в добавлении к запросу описания процесса решения задачи помимо самих задачи и решения. Использование такой подсказки в SayCan [13] позволяет улучшить работу модели для задач, где используется отрицание или процесс рассуждения. Ограничение такого подхода заключается в том, что, как показано в [14], улучшения демонстрируются только для БЯМ с более чем 100 миллиардами параметров.

В архитектуре ProgPromt [15] основная идея заключается в представление запроса в виде Python-подобного кода. Запрос состоит из описания доступных действий в виде импортирования соответствующих программных модулей, списком с перечислением доступных объектов на сцене, примерами задачи и их выполнения в виде программных модулей. Использование такого вида запроса обосновывается тем, что БЯМ, такие как GPT-3 с 175 млрд параметров [5], обучаются в том числе и на большом количестве данных из открытых репозиториев программного кода. Похожий подход используется в модели CaP [16], которая генерирует стратегию агента. Отличительной чертой является то, что получаемые программы являются полноценным исполняемым программным кодом и позволяют организовать иерархичность выполнения самих программ.

Некоторые работы, например Socratic Model [17] и Inner Monologue [18], предлагают не конкретную модель, а подход к построению и объединению множества моделей. Так, в Socratic

Model [17] предлагается использовать предобученные модели, использующие разные модальности (звук, текст, изображение и др.). Предлагается объединять такие модели в системы, которые способны решать задачи, выходящие за рамки задач каждой отдельной модели. Это достигается за счет создания интерфейса обмена данными между моделями. В качестве примера робототехнической задачи предлагается планирование перемещения объектов на столе. Используется симулятор Pybullet [19] для детектирования объектов на сцене. Для составления описания по изображению используется подход ViLD [20], после чего описание сцены в виде запроса передается в БЯМ для генерации плана по аналогии с Zero-Shot Planners [7] и SayCan, далее план выполняется CLIPort-подобной стратегией [21, 22]. Запрос состоит из описания сцены в формате перечисления объектов python-подобным списком, примеров формулировки задач на естественном языке и их выполнения в виде псевдокода.

В архитектуре Inner Monologue [18] предлагается использовать обратную связь в виде текста, полученную либо от среды (описание сцены, успешность выполнения действия), либо от пользователя (уточнение необходимого действия агента). При этом, сохраняя общий подход, в зависимости от задачи для реализации используются разные модели. Основная идея заключается в итеративном добавлении обратной связи от среды в виде текста во входной запрос, используемый для планирования БЯМ. При манипулировании с объектами (в среде с виртуальным или реальным столом) используется запрос с описанием сцены и примерами задач. Для выполнения задачи на реальной кухне запрос форматируется в виде диалога пользователя и воплощенного агента.

В работе LM-Nav [23] используется подход, попадающий под определение Socratic Models [17], когда для задачи навигации по тексту и изображению используются предобученные отдельно модели, соединенные в общую систему. БЯМ GPT-3 [5] используется для генерации последовательности текстовых ориентиров на основе инструкций на естественном языке, визуально-языковая модель сопоставляет текстовые ориентиры с изображениями, получаемыми агентом [24], а модель навигации по изображению [25] строит и выполняет план перемещения агента. В качестве запроса для БЯМ используются три примера извлечения текстовых ориентиров.

3. КЛАССИФИКАЦИЯ БЯМ ДЛЯ ЗАДАЧИ ПЛАНИРОВАНИЯ

Классификация рассмотренных подходов по типам используемых запросов, средам тестирования и применяемых БЯМ представлена в табл. 1. Обобщая данные, представленные в табл. 1, необходимо заметить, что в общем виде запрос для языковой модели может состоять из следующих частей:

1. Описания сцены, которое заключается или в простом перечислении доступных объектов, или дополняется приведением свойств этих объектов;

2. Перечисления доступных для воплощенного агента действий;

3. Примеров задач, поставленных перед воплощенным агентом;

4. Примеров выполнения поставленных задач.

При этом сам запрос может выражаться или простым текстом, или в виде диалога. Также помимо запроса на естественном языке возможно использовать запросы, представляющие собой или псевдокод, или выполняемый программный код, например на языке Python.

Необходимо отметить, что задача подбора правильного запроса является непростой и на результат могут влиять такие, казалось бы, незначительные изменения, как нумерация списка действий плана и добавления символов переноса строки (см. примеры в SayCan [13]).

Все рассмотренные подходы могут быть описаны общей архитектурой использования БЯМ для задачи планирования действий воплощенного агента, представленной на рис. 1.

На первом этапе, на основе инструкции на естественном языке, описывающей задачу для воплощенного агента, формируется запрос для БЯМ. В простейшем случае запрос состоит из примеров задач с планами выполнения в терминах, доступных для воплощенного агента инструкций [7, 13, 23]. Более сложная структура запроса может включать дополнительную информацию о среде, например, перечисление присутствующих объектов и их свойств [10, 12, 15–18] или доступных действий агента [15, 16]. Далее запрос поступает в БЯМ, которая итеративно генерирует план поведения. Стоит заметить, что в основном запрос формулируется на естественном языке [7, 10, 12, 13, 17, 18, 23], но существуют подходы, использующие для этого псевдокод [16, 17] или программный код [15]. На втором этапе агент выполняет полученный план. Такая постановка подразумевает, что план поведения генерируется полностью до начала выполнения в среде и не модифицируется в процессе выполнения. Это может привести к ситуации, когда агент застревает на одном из этапов выполнения плана, что, в свою очередь, может привести к невыполнению исходной задачи. Решением этой проблемы может быть использование обратной связи от среды (пунктирная стрелка на рис. 1) на каждой итерации генерирования следующего действия агента. В качестве обратной связи может использоваться информация о возможности выполнения конкретного действия [13] или отчет о кор-

Таблица 1. Сравнения алгоритмов для задач воплощенного искусственного интеллекта на основе предобученных БЯМ

Алгоритм	Языковая модель	Среда	Робот. реализация	Тип запроса	Навигация	Взаимод. со средой
Zero-Shot Planners [7]	GPT-3 175B [5] Codex 12B [29]	VirtualHome [8]	–	Примеры задач и их выполнения	+	+
G-PlanET [10]	TaPEX [30]	ALFRED [11] (модификация)	–	Описание сцены, описание задачи	–	–
EA-APG [12]	GPT-3 [5]	VirtualHome [8]	–	Описание сцены, примеры задач и их выполнения	+	+
SayCan [13]	PALM 540B [4]	Естественная среда (кухня)	Everyday Robots	Примеры задач их решения, сформулированные в виде диалога	+	+
ProgPromt [15]	GPT-3 175B [5]	VirtualHome [8]	–	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнением	+	+
Socratic [17]	GPT-3 175B [5]	Pybullet [19]	–	Описание сцены, примеры задач и их выполнение на псевдокоде	–	+
Inner Monologue [18]	InstructGPT [31]	Pybullet [19]	–	Описание сцены, примеры задач и их выполнения	–	+
	InstructGPT [31]	Естественная среда (стол)	Everyday Robots	Описание сцены, примеры задач и их выполнения	–	+
	PALM 540B [4]	Естественная среда (кухня)	Everyday Robots	Примеры задач и их выполнения в виде диалога	+	+
CaP [16]	Codex [29] code-davinci-002	Естественная среда (рисование на белой доске)	UR5e	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнением	+	–
	Codex [29] code-davinci-002	Естественная среда (стол)	UR5e	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнением	+	–
	Codex [29] code-davinci-002	Естественная среда (кухня)	Everyday Robots	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнением	+	+
LM-Nav [23]	GPT-3 [5]	Естественная среда (улица)	Clearpath Jackal UGV	Примеры задач и их выполнения	+	–

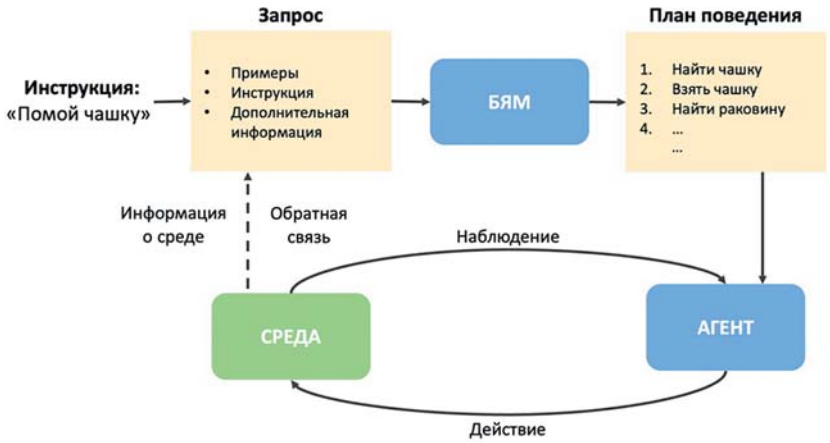

Рис. 1. Общая архитектура использования больших языковых моделей (БЯМ) для задачи построения плана поведения воплощенного агента.

ректном выполнении действия или об изменении состояния объекта [18].

4. ЗАКЛЮЧЕНИЕ

Рассмотренные подходы использования БЯМ для планирования поведения демонстрируют неплохие результаты как в виртуальных средах, так и при имплементации на робототехнических платформах. Тем не менее количественное сравнение этих работ осложнено тем, что в них используются (за исключением нескольких работ) различные среды. При этом существуют и хорошо известны задачи с установленными метриками качества и таблицами сравнений, на которых тестируются воплощенные интеллектуальные агенты, например ALFRED [11] и TEACh [26] для следования инструкциям на естественном языке, RoomR [27] и Benchbot [28] для перестановки объектов, и другие. К сожалению, пока в этих бенчмарках большинство указанных работ не представлены, что во многом объясняется сложностью организации запросов в разнообразных средах с большим количеством объектов и действий.

Перспективным будущим направлением работ являются, во-первых, усложнение и обучение обратной связи для определения качества выдаваемых ответов БЯМ, и, во-вторых, модификация процесса обучения самих БЯМ, чтобы добавить регуляризатор, моделирующий реалистичностость генерируемых ответов, в функцию потерь всей языковой модели.

СПИСОК ЛИТЕРАТУРЫ

1. *Min S.Y. et al.* Film: Following instructions in language with modular methods //arXiv preprint arXiv:2110.07342. 2021.
2. *Liu H. 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. *Devlin J. et al.* Bert: Pre-training of deep bidirectional transformers for language understanding //arXiv preprint arXiv:1810.04805. 2018.
4. *Chowdhery A. et al.* Palm: Scaling language modeling with pathways //arXiv preprint arXiv:2204.02311. 2022.
5. *Brown T. et al.* Language models are few-shot learners // Advances in neural information processing systems. 2020. Т. 33. С. 1877–1901.
6. *Wei J. et al.* Finetuned language models are zero-shot learners //arXiv preprint arXiv:2109.01652. 2021.
7. *Huang W. et al.* Language models as zero-shot planners: Extracting actionable knowledge for embodied agents // arXiv preprint arXiv:2201.07207. 2022.
8. *Puig X. et al.* Virtualhome: Simulating household activities via programs //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. С. 8494–85.
9. *Reimers N., Gurevych I.* Sentence-bert: Sentence embeddings using siamese bert-networks //arXiv preprint arXiv:1908.10084. 2019.
10. *Lin B.Y. et al.* On Grounded Planning for Embodied Tasks with Language Models // arXiv preprint arXiv:2209.00465. 2022.
11. *Shridhar M. 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. С. 10740–10749.
12. *Gramopadhye M., Szafir D.* Generating Executable Action Plans with Environmentally-Aware Language Models //arXiv preprint arXiv:2210.04964. 2022.
13. *Ahn M. et al.* Do as i can, not as i say: Grounding language in robotic affordances //arXiv preprint arXiv:2204.01691. 2022.
14. *Wei J. et al.* Chain of thought prompting elicits reasoning in large language models //arXiv preprint arXiv:2201.11903. 2022.

15. *Singh I. et al.* ProgPrompt: Generating Situated Robot Task Plans using Large Language Models //arXiv preprint arXiv:2209.11302. 2022.
16. *Liang J. et al.* Code as policies: Language model programs for embodied control //arXiv preprint arXiv:2209.07753. 2022.
17. *Zeng A. et al.* Socratic models: Composing zero-shot multimodal reasoning with language //arXiv preprint arXiv:2204.00598. 2022.
18. *Huang W. et al.* Inner monologue: Embodied reasoning through planning with language models //arXiv preprint arXiv:2207.05608. 2022.
19. *Coumans E., Bai Y.* Pybullet, a python module for physics simulation for games, robotics and machine learning. GitHub Repository – 2016.
20. *Gu X. et al.* Open-vocabulary object detection via vision and language knowledge distillation //arXiv preprint arXiv:2104.13921. 2021.
21. *Shridhar M., Manuelli L., Fox D.* Cliport: What and where pathways for robotic manipulation //Conference on Robot Learning. PMLR, 2022. С. 894–906.
22. *Zeng A. et al.* Transporter networks: Rearranging the visual world for robotic manipulation //arXiv preprint arXiv:2010.14406. 2020.
23. *Shah D. et al.* Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action // arXiv preprint arXiv:2207.04429. 2022.
24. *Radford A. et al.* Learning transferable visual models from natural language supervision //International Conference on Machine Learning. PMLR, 2021. С. 8748–8763.
25. *Shah D. et al.* Ving: Learning open-world navigation with visual goals //2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021. С. 13215-13222.
26. *Padmakumar A. et al.* Teach: Task-driven embodied agents that chat //Proceedings of the AAAI Conference on Artificial Intelligence. 2022. Т. 36. №. 2. С. 2017–2025.
27. *Weihs L. et al.* Visual room rearrangement //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. С. 5922–5931.
28. *Talbot B. et al.* Benchbot: Evaluating robotics research in photorealistic 3d simulation and on real robots // arXiv preprint arXiv:2008.00635. 2020.
29. *Chen M. et al.* Evaluating large language models trained on code //arXiv preprint arXiv:2107.03374. 2021.
30. *Liu Q. et al.* Tapex: Table pre-training via learning a neural sql executor //arXiv preprint arXiv:2107.07653. 2021.
31. *Ouyang L. et al.* Training language models to follow instructions with human feedback //arXiv preprint arXiv:2203.02155. 2022.