DOI: https://doi.org/10.26565/2304-6201-2025-65-07

УДК (UDC) 004.8

Omelchenko Ihor Valeriiovych *PhD student, Department of Mathematical Modeling and Data Analysis Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine, 61022*
e-mail: ihor.v.omelchenko@gmail.com;
https://orcid.org/0009-0007-4474-4916

Strukov Volodymyr Mykhailovych *PhD in Technical Sciences, Associate Professor; Head of the Department of Mathematical Modeling and Data Analysis*
Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine, 61022
e-mail: volodymyr.strukov@karazin.ua;
http://orcid.org/0000-0003-4722-3159

On the impact of prompts on agent performance in a virtual environment

Relevance: Currently, it is promising to study the application of language models in decision-making tasks. It is possible to use pre-trained language models that demonstrate skills in working with arbitrary text, solving logical tasks, and are able to learn from text examples. Such language models are able to solve new tasks that are presented in text form.

Goal: The goal is to conduct a study of the influence of various language instructions (prompts) on the functioning of an agent in a virtual environment. The agent functions on the basis of a pre-trained language model.

Research methods: To perform the study, a Minigrid virtual environment and pre-trained language models were used, a software agent was created based on the language model, a set of language instructions was created using such methods as zero-shot learning, few-shot learning, and others. The effectiveness of the agent's functioning is estimated using the following numerical values: total reward in the environment, episode duration, number of language model calls. Experiments were conducted to train and test a software agent in a virtual environment. Numerical and statistical results of experiments were collected.

Results: Differences in the functioning of the agent were revealed when using different methods of designing language instructions. Language instructions that contain examples of solving tasks lead to better results than those that present the task in imperative form. An improvement in agent performance was also demonstrated upon the addition of a default action plan. Adding episodic memory into the agent further enhanced performance in specific cases.

Conclusions: In this work, we considered a software agent based on a pre-trained language model that solves the decision-making problem in a virtual environment.

Keywords: *machine learning, deep learning, artificial neural network, language model, prompt, decision-making, reinforcement learning, PPO, agent, virtual environment, minigrid.*

How to quote: I. Omelchenko, and V. Strukov, “On the impact of prompts on agent performance in a virtual environment”, *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 65, pp. 83-91, 2025. https://doi.org/10.26565/2304-6201-2025-65-07

Як цитувати: . Omelchenko I., Strukov V. On the impact of prompts on agent performance in a virtual environment. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2025. вип. 65. С.83-91.
https://doi.org/10.26565/2304-6201-2025-65-07

1. Вступ

In recent years, the scientific community has been particularly interested in language models (LM) [1]. Such models are organized as deep neural networks that solve the problem of determining the next token (word fragment) in a text sequence. When trained on large and high-quality text data, language models demonstrate the ability to process arbitrary texts, understand different languages and solve logical and mathematical problems. Such abilities can be used to solve the problem of decision-making in the environment [2].

The decision-making problem can be represented as a hierarchy of actions [3], where higher levels consist of a sequence of lower-level actions. Such high-level actions can be considered as an agent's plan of actions.

In recent works [4], the possibilities of integrating language models into software agents were considered. The influence of the structure of language instructions on the quality of agent functioning is not completely explored.

These works consider agents that have a three-component structure that includes a high-level planner, an execution module and a data representation adapter [5]. This structure allows the system to be used in different environments and reduces the number of necessary modifications. At the same time, the language model remains unchanged, while one changes the language instructions that are specific to a particular environment and task.

Language models are able to analyze data received as a feedback from the environment in response to actions selected by the language model [6]. Language models are able to generate an internal monologue that improves the quality of task performance. This system was applied to solving tasks of moving objects by a robotic system in a physical environment in response to a user request.

In this work, we investigate the influence of language instructions on the quality of agent performance values. We start with the results of the work [4], which we are planning to improve by considering various language instructions. In this work we use a two-dimensional discrete virtual environment and a software agent that is able to perform actions in the environment. A set of approaches to creating language instructions is investigated, such as zero-shot and few-shot in-context learning supplemented with default plan and memory.

2. Problem formulation

2.1 Agent

In the study, we use a software agent similar to one in [4], which consists of three components: Planner, Mediator, Actor. The Planner component includes a pre-trained language model and performs the task of generating a plan of abstract actions based on the observation of the state of the environment. This component accepts data as input and returns data in a text representation as output. The Mediator component is used to convert data between a text representation and a representation specific to observations and actions of the environment. Abstract actions generated by the Planner component are passed to the input of the Actor component, which converts abstract actions into a sequence of specific actions that will be performed in the environment. In addition, the Actor decides when the plan is completed or when it is necessary to call the Planner and generate a new action plan.

During the training of the agent, we optimize the Actor component which decides when it is necessary to generate a new plan. At the same time, the language model, which is part of the Planner component, remains unchanged.

2.2 Environment

In this work, we use the Minigrid virtual environment. This environment provides a set of instruments for creating specific tasks that include navigating two-dimensional mazes and interacting with objects in the maze. Our study uses a modified version of the Minigrid environment, which limits the area visible to the agent, so that the agent needs to explore the environment at first to learn about its structure and objects in the environment.

The environment usually contains walls that restrict the agent's movement, doors that can be closed, limiting visibility and requiring the use of a key to open them, boxes that can contain other objects, circles that serve as obstacles which the agent can move by itself. The objects are distinguished by color, which adds variability to the task formulation and adds combinatorial complexity to the space of possible solutions.

To simplify the task we consider the environment without obstacles. So, the task is to find the key and open the door in the smallest possible number of steps. Some examples of this environment are given in Fig. 1.

The agent interacts with the environment during episodes. At the beginning of an episode, the environment is initialized with a specific task and the agent is placed in a random location in the maze. Then, the agent can perform a sequence of actions that can lead to the successful completion of the task or to reaching the step limit, which is 100 steps, after which the episode is considered complete.

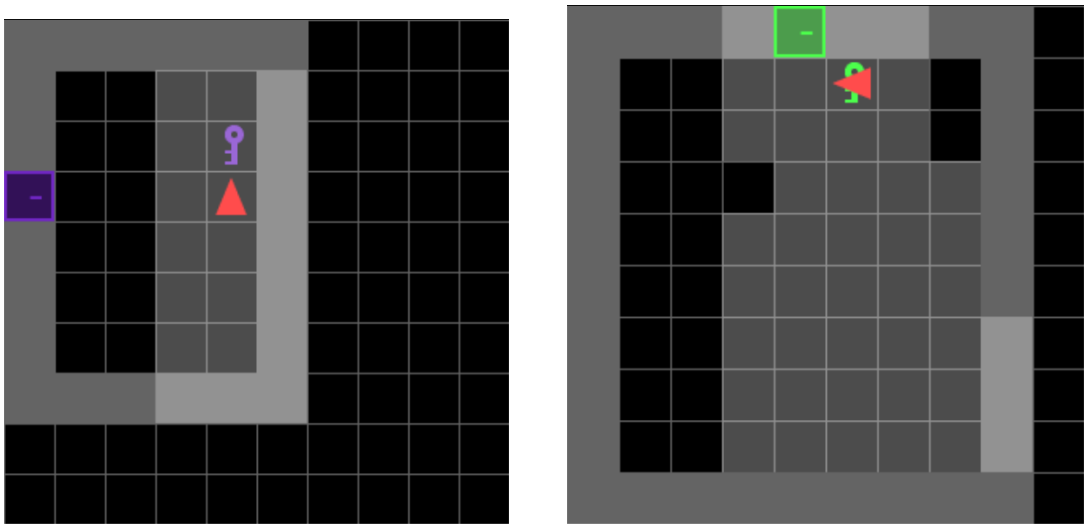

Figure 1: Examples of the environment
Рисунок 1: Приклади оточення

3. Prompts

Modern prompt creation techniques such as zero-shot and few-shot prompt [7] were used to test the agent. Basic versions of prompts that are capable of creating only one action instead of a sequence of actions were also tested. In addition, memory for one episode was tested.

The zero-shot prompt technique consists in passing the task description to the language model without examples of solutions. Instead, the few-shot prompt technique includes a set of examples of correct task solutions, so that in this case, the language model generalizes the examples and applies a general scheme for solving the task.

The default plan prompt was tested, in which case the model received an example of solving the task, which with a small number of modifications could be transformed into a solution for a specific situation.

Memory lasting one episode was also used. With memory turned off, the agent receives only observations of the current state of the environment. In this case, the agent does not take into account its own previous mistakes and cannot learn from its own experience of interacting with the environment. In the case of memory presence, the agent receives a history of interactions with the environment from the beginning of the current episode.

4. Training

During the agent training, one of the agent's neural networks was optimized. This neural network functions as the communication policy. This component decides when to generate a new action plan, which leads to the language model call. The language model execution requires significant computing power and takes a significant amount of time compared to other elements of the system, so reducing the number of language model calls leads to a decrease in the average duration of one iteration. The communication policy was trained using the Proximal Policy Optimization (PPO) reinforcement learning method [8]. As a result of training, the communication policy called the language model only in cases when it was necessary to generate a new plan. At the same time, the pre-trained language model remained unchanged, it was not fine-tuned.

Training lasted for 1000 iterations, where each iteration corresponded to one episode of the interaction with the environment. From the Fig. 2 we see that the average reward during the episode increased to an average value of 0.6 as training progressed. The Fig. 3 shows the duration of the training episodes. We note that an episode cannot last more than 100 iterations, so after 100 iterations in one episode the agent is considered to have failed the task. From this figure, we see that the average value fluctuates around an episode duration of 30 steps, which indicates that the agent is able to successfully complete the task in most cases in less than 100 iterations. The Fig. 4 shows the average number of language model calls for each episode during training. From this figure, we can see that the number of language model calls drops rapidly and reaches a value of 3 calls per episode by the end of training.

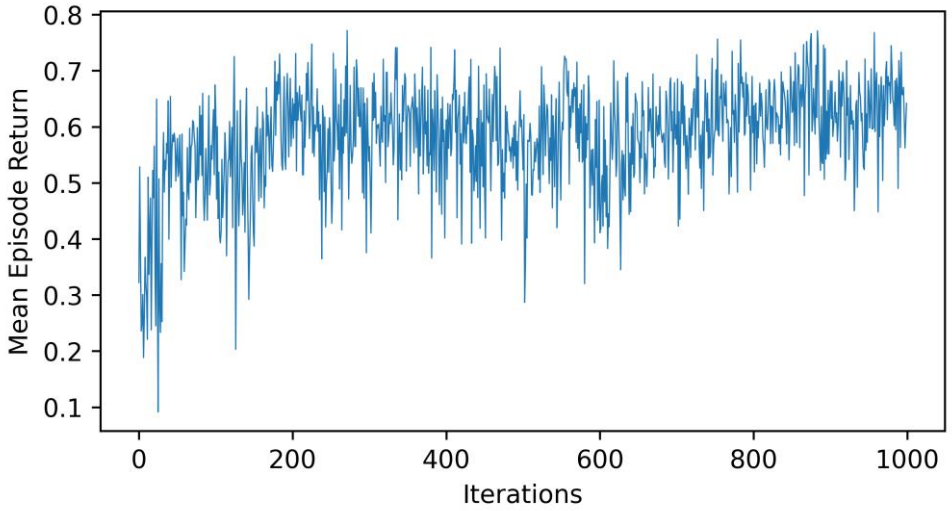

Figure 2: The dependence of mean episode return on number of training iterations
Рисунок 2: Залежність середнього результату епізоду від кількості ітерацій навчання

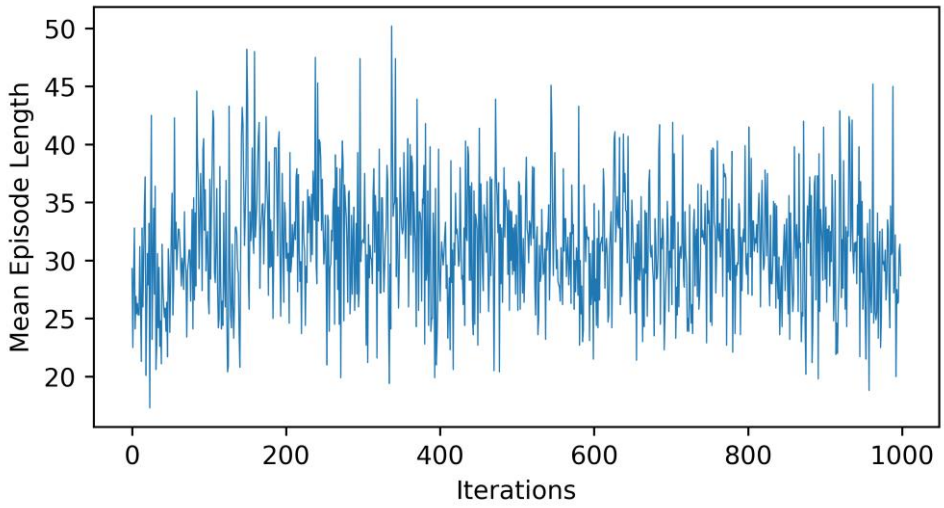

Figure 3: The dependence of mean episode length on number of training iterations
Рисунок 3: Залежність середньої тривалості епізоду від кількості ітерацій навчання

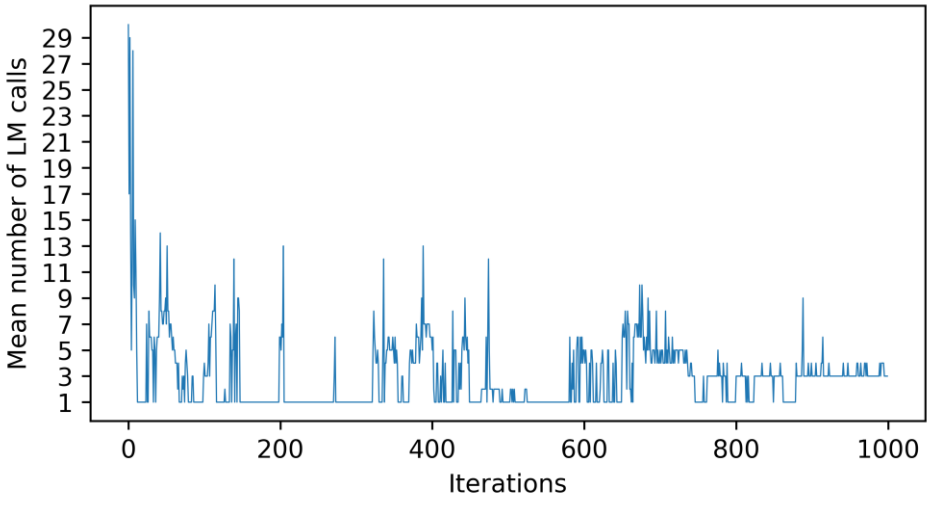

Figure 4: The dependence of mean number of LM calls on number of training iterations
Рисунок 4: Залежність середньої кількості викликів LM від кількості ітерацій навчання

5. Testing

A trained agent was used to test various prompting techniques. Testing was performed on 100 examples of the "Simple Door Key" Minigrid environment, each example of the environment corresponds to a unique random value of initialization seed. During testing, the following metrics were calculated: return, return without communication penalties, number of language model calls, episode duration.

The Fig. 5 shows the quartile return values for different prompts with memory enabled and disabled. From this figure, we see that for the original, few-shot, few-shot default plan prompts, the use of memory does not lead to noticeable improvements. In the case of the "few shot one step" prompt, memory allows the agent to remember the state of the environment between individual iterations, in which the agent can choose only one action. A significant improvement contributed by the memory usage is shown in the case of the "zero shot default plan" prompt, which is explained by the fact that the agent gets access to previous actions in the current episode, which helps to correctly apply the default plan.

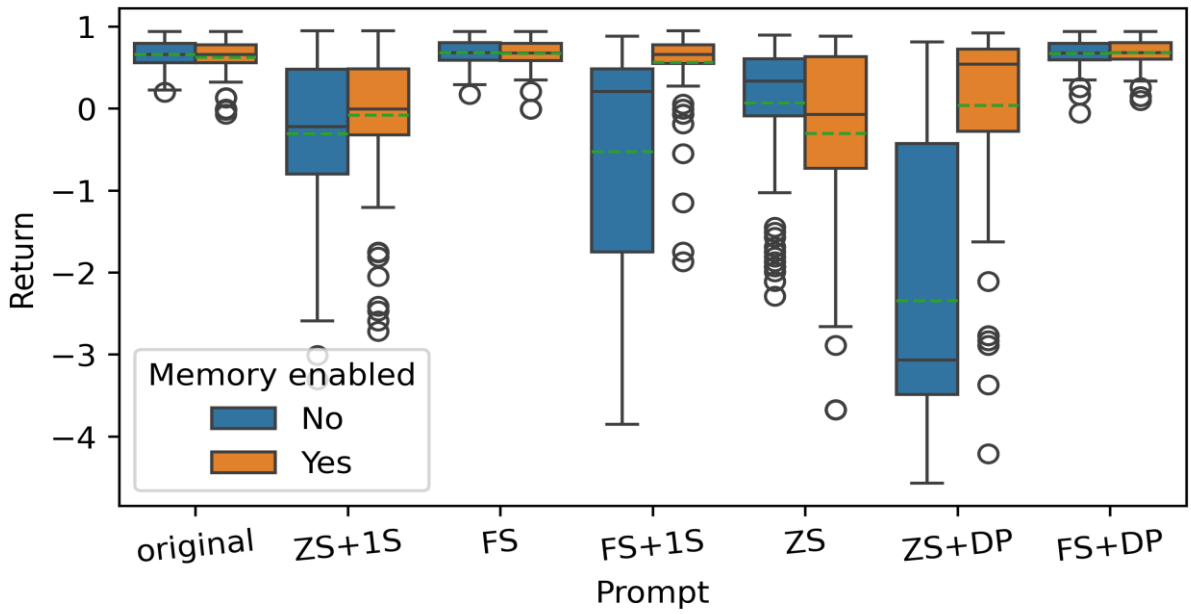

Figure 5. The return value for different prompts depending on memory being enabled or disabled: "original" means prompt from [4], "ZS" is for zero-shot prompting technique, "FS" is for few-shot prompting technique, "1S" is for one-step planning, "DP" is for default plan

Рисунок 5. Результат для різних промптів залежно від увімкнення або вимкнення пам'яті: «original» означає промпт з [4], «ZS» — техніку промптінгу zero-shot, «FS» — техніку промптінгу few-shot, «1S» — однокрокове планування, «DP» — план за замовчуванням.

The Fig. 6 shows the dependence of the penalty for the language model call on the type of prompt and the availability of memory for the agent. A high penalty value corresponds to frequent calls of the language model to rebuild the plan. From the data, we see that the "few-shot one step" prompt without memory often calls the language model, but with the memory enabled, the situation improves significantly. A similar situation is observed in the case of the "zero-shot default plan" and "zero-shot one step" prompts: when memory is enabled, the agent calls the language model less often. In the case of the "zero shot" prompt, enabling memory leads to more frequent language model calls and a higher penalty. In general, we see that prompts of the "few-shot" category receive a significantly lower penalty than "zero-shot" prompts.

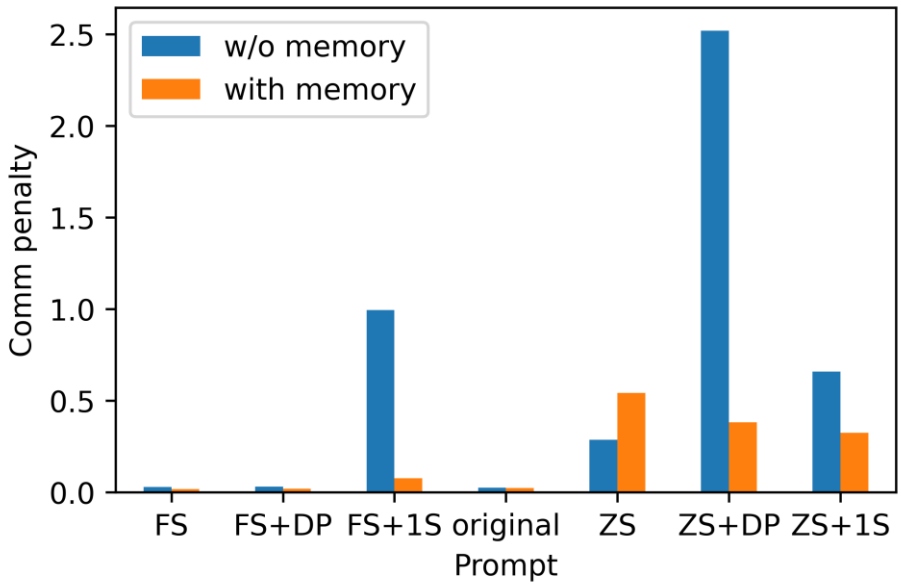

Figure 6. The communication penalty value for different prompts (as in Fig. 5) for memory being enabled or disabled

Рисунок 6. Значення штрафу за комунікацію для різних промптів (як на рис. 5) для пам'яті, що ввімкнена або вимкнена.

The Fig. 7 shows a comparison of the return for different prompts with memory enabled with a percentage improvement compared to the original prompt [4]. Correcting the prompt grammar and adding a default plan gave an improvement of 7.34% and 8.97%, respectively. In the case where the agent could choose only one action at a time, the degradation was -10.5%. Prompts in the zero shot category showed a significant drop in the total reward, up to negative values. These results show that examples of solving the task significantly increase the return received by the agent.

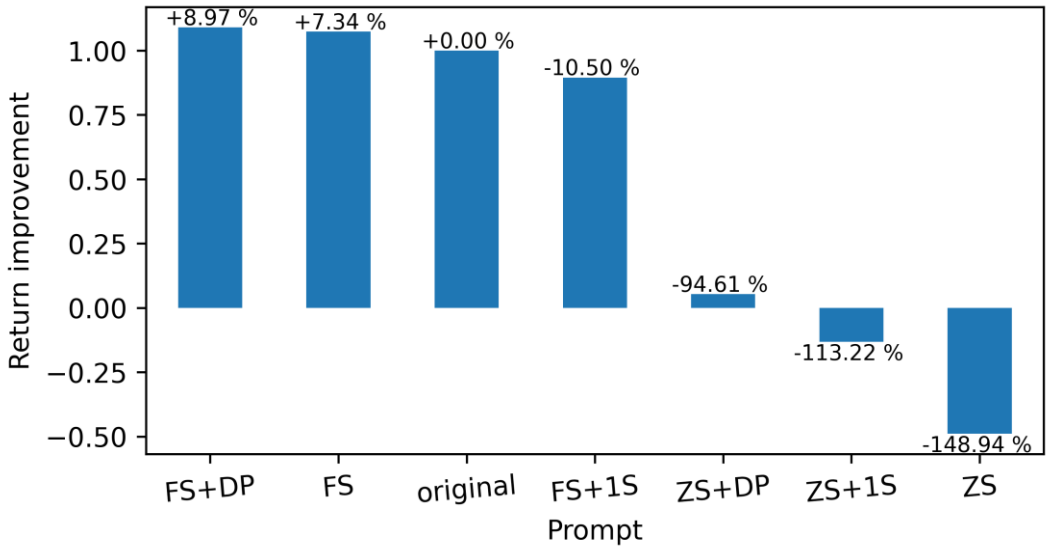

Figure 7. The relative return improvement value comparing to original case from [4] for different prompts (as in Fig. 5)

Рисунок 7. Відносне покращення результату порівняно з початковим випадком з [4] для різних промптів (як на рис. 5)

The average episode duration in Fig. 8 also demonstrates the advantage of few-shot prompts over the original prompt and prompts without examples (zero-shot).

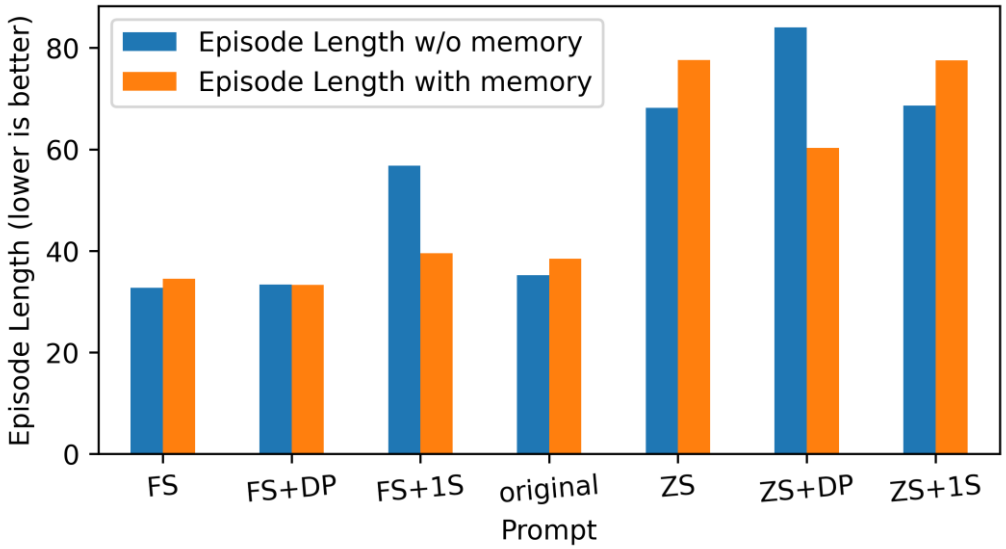

Figure 8. The episode length value for different prompts (as in Fig. 5) for memory being enabled or disabled
Рисунок 8. Значення тривалості епізоду для різних промптів (як на рис. 5) для пам'яті, що ввімкнена або вимкнена.

The Fig. 9 shows the dependence of the number of language model calls on the prompt. In the absence of examples (zero-shot), the agent calls the language model much more often than in cases where the agent has examples available (few-shot).

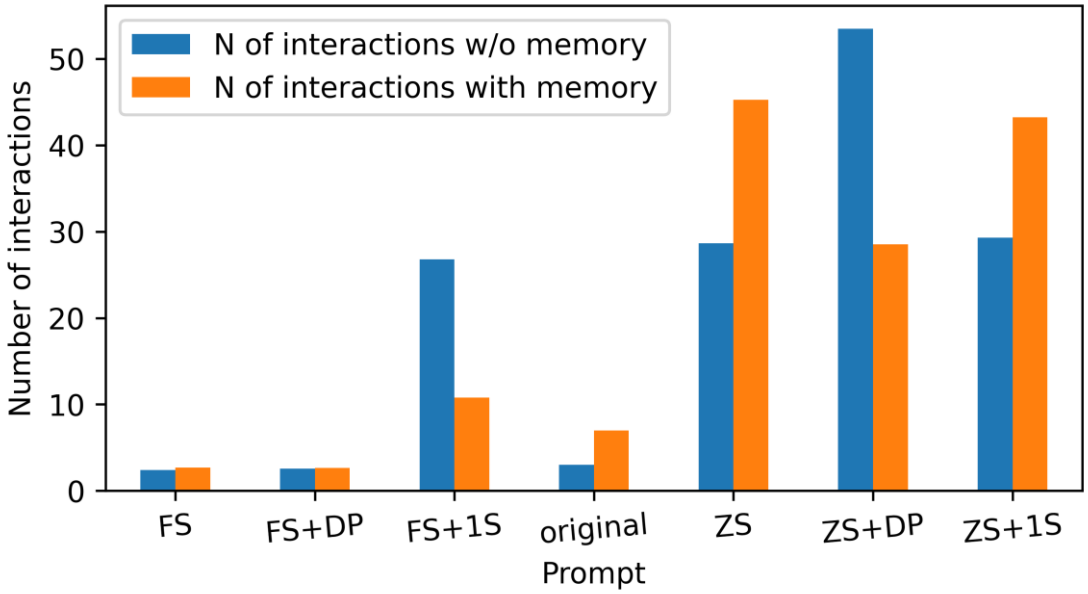

Figure 9. The number of interactions value for different prompts (as in Fig. 5) for memory being enabled or disabled
Рисунок 9. Кількість взаємодій для різних промптів (як на рис. 5) для увімкненої або вимкненої пам'яті

The Fig. 10 shows the dependence of the number of language model calls depending on the prompt, arranged by the increasing number of calls. We can see that the best result was achieved by the agent with the prompt containing examples of solving the task and the default plan.

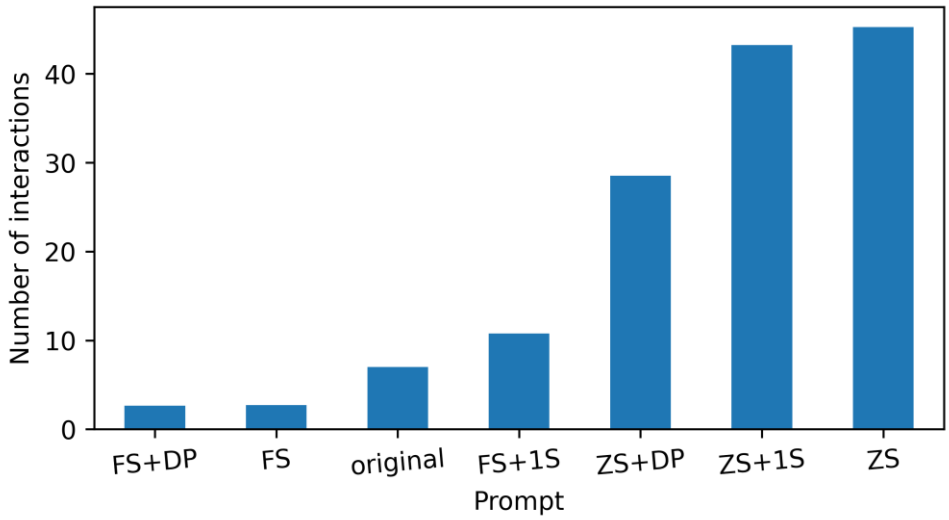

Figure 10. The number of interactions for different prompts (as in Fig. 5) with memory being enabled. The less value corresponds to better result

Рисунок 10. Кількість взаємодій для різних промптів (як на рис. 5) з увімкненою пам'яттю. Менше значення відповідає кращому результату.

6. Conclusions

So, we conducted the research of the agent built on the basis of a language model, which solves the problem of decision-making in the environment. For training and testing, the Minigrid "Simple Door Key" environment was used, which sets the task of navigation in a two-dimensional environment and interaction with objects in it. In this environment, the agent had to find a hidden key and open the door with it. Among the agent components, only the communication policy component was trained, and the language model remained unchanged. As a result of training, the agent reduced the number of language model calls.

The trained agent was tested on 100 instances of the environment. The resulting average metrics demonstrate a significant advantage of prompts with examples (few-shot) over prompts without examples (zero-shot). Some improvement is made by adding a default plan. It is also shown that the agent achieves better results when generating a sequence of actions compared to the situation when the agent can choose only one action at a time. In addition, it can be concluded that it is important how correctly the prompts are written (in a grammatical sense), this is demonstrated by the example of improving the performance of the "few-shot" prompt compared to the "original" prompt.

Therefore, when choosing a technique for creating prompts for agents based on language models, it is worth considering that the presence of examples of solving the task significantly improves the agent's performance. The presence of a default plan has a smaller impact. The presence of memory helps in certain situations, such as when the agent is restricted to choosing only one action at a time. The absence of examples of solving the task and choosing only one action per step can lead to the inability of the agent to complete the task.

In the future, it is of interest to use more complex prompting techniques, use better language models, study the agent's performance in more complex environments, and use agent architectures which allow training both the language model and options at the same time.

REFERENCES

1. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762
2. L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen, "A survey on large language model based autonomous agents," Frontiers of Computer Science, vol. 18, no. 6, Mar. 2024. [Online]. Available: http://dx.doi.org/10.1007/s11704-024-40231-1

3. R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," Artificial Intelligence, vol. 112, no. 1, pp. 181--211, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370299000521
4. B. Hu, C. Zhao, P. Zhang, Z. Zhou, Y. Yang, Z. Xu, and B. Liu, "Enabling intelligent interactions between an agent and an LLM: A reinforcement learning approach," Reinforcement Learning Journal, vol. 3, pp. 1289--1305, 2024. [Online]. Available: https://rlj.cs.umass.edu/2024/papers/RLJ_RLC_2024_161.pdf
5. I. Dasgupta, C. Kaeser-Chen, K. Marino, A. Ahuja, S. Babayan, F. Hill, and R. Fergus, "Collaborating with language models for embodied reasoning," 2023. [Online]. Available: https://arxiv.org/abs/2302.00763
6. W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," 2022. [Online]. Available: https://arxiv.org/abs/2207.05608
7. Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, and Z. Sui, "A survey on in-context learning," 2024. [Online]. Available: https://arxiv.org/abs/2301.00234
8. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

Омельченко Ігор Валерійович *Аспірант, кафедра математичного моделювання та аналізу даних Харківський національний університет ім. В.Н. Каразіна. Площа Незалежності, 4, Харків, Харківська область, 61022*

Струков Володимир Михайлович *к.т.н., доцент; завідувач кафедри математичного моделювання та аналізу даних Харківський національний університет ім. В.Н. Каразіна. Площа Незалежності, 4, Харків, Харківська область, 61022*

Дослідження впливу мовних інструкцій на якість роботи програмного агента в віртуальному середовищі

Актуальність. Наразі є перспективним дослідження застосування мовних моделей в задачах прийняття рішень. Можливо застосувати попередньо навчені мовні моделі, які демонструють навички роботи з довільним текстом, вирішення логічних завдань та які здатні навчатися з текстових прикладів. Такі мовні моделі здатні вирішувати нові завдання, які представлені у текстовому вигляді.

Мета. Провести дослідження впливу різних мовних інструкцій (англ. prompt) на функціонування агента в віртуальному середовищі. Агент функціонує на основі попередньо-навченої мовної моделі.

Методи дослідження. Для виконання дослідження було використано віртуальне середовище, попередньо навчені мовні моделі, створено програмного агента на основі мовної моделі, створено набір мовних інструкцій з застосуванням таких методів як zero-shot learning, few-shot learning та інших. Ефективність функціонування агента оцінюється за допомогою таких числових величин, сумарна нагорода в середовищі, тривалість епізоду, кількість викликів мовної моделі. Проведено експерименти з навчання та тестування програмного агента у віртуальному середовищі. Зібрано чисельні та статистичні результати експериментів.

Результати. Виявлено відмінність у функціонуванні агента при застосуванні різних методів дизайну мовних інструкцій. Мовні інструкції, які містять приклади вирішення завдань, призводять до кращих результатів, ніж ті, що подають завдання в імперативній формі. Також продемонстровано покращення роботи агента при додаванні плану дій за змовчуванням. Додавання агенту пам'яті, що триває один епізод покращило результати в окремих випадках.

Висновки. В роботі було розглянуто програмного агента на основі попередньо-навченої мовної моделі, який вирішує задачу прийняття рішень в віртуальному середовищі.

Ключові слова: *машинне навчання, глибоке навчання, штучна нейронна мережа, мовна модель, промпт, прийняття рішень, навчання з підкріпленням, PPO, агент, віртуальне середовище, minigrid.*

Надійшла у першій редакції 14.02.2025, в останній - 30.03.2025.
The first version has been received on 14.02.2025, the final version - on 30.03.2025.