

Selective Perception: Learning Concise State Descriptions for Language Model Actors

Kolby Nottingham and Yasaman Razeghi and Kyungmin Kim and JB Lanier and Pierre Baldi and Roy Fox and Sameer Singh

University of California Irvine

{knotting,yrazeghi,kyungk7,jblanier,pfbaldi,royf,sameer}@uci.edu

Abstract

The latest large language models (LMs) support increasingly longer contexts. While this trend permits using substantial amounts of text with SOTA LMs, requiring these large LMs to process potentially redundant or irrelevant data needlessly increases inference time and cost. To remedy this problem, we propose BLINDER, a method that leverages a small finetuned LM to sample the minimal set of input features that maximizes the performance of a downstream LM. BLINDER trains an LM with a value head to estimate the likelihood of optimal outputs from a downstream LM given an input. We evaluate BLINDER on embodied decision making tasks with notoriously verbose state descriptions: NetHack and robot planning. BLINDER reduces the length of LM actor input by 87% and 99% while improving task success rates by 158% and 54% on NetHack and robot planning respectively which represents substantial inference cost savings while actually increasing performance.

1 Introduction

Large language models (LMs) continue to scale in both number of parameters and supported context length. This trend has given rise to LMs that are able to solve complex problems across many large documents. However, this comes at a price (often literally) as context length contributes directly to the time and compute cost of LM inference.

While previous work has optimized fewshot examples (Rubin et al., 2022; Wang et al., 2023) or prompt instructions (Shi et al., 2022; Fernando et al., 2023), the remaining input is left unmodified. Also, previous work has focused on improving performance rather than conciseness. The goal of this work is to develop a method for cheaply reducing input length while maintaining or improving downstream task performance. We choose to explore learning concise inputs in the context of

embodied decision making which exacerbates the problem of input length with verbose scene information (Huang et al., 2022b; Singh et al., 2022).

Embodied decision making tasks evaluate LMs as actors by generating actions given a task and state description. While an exhaustive list of state features is often used as a description for an LM actor (Huang et al., 2022b), such inputs can contain many features that are irrelevant to or distract from the current task. LMs are adept summarizers (Liu and Lapata, 2019; Stiennon et al., 2020a), and an LM summarizer could be used to summarize the input to a much larger LM for a difficult downstream task. However, this comes with an obvious trade off as better summaries will come from larger LMs that do not satisfy our primary objective: reducing overall compute requirements.

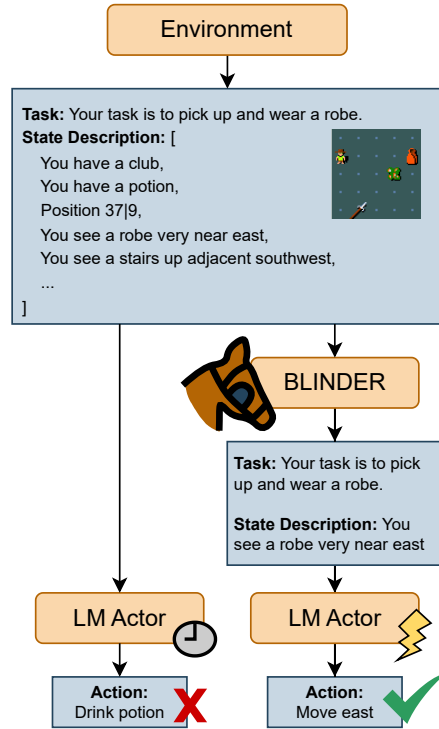


Figure 1: BLINDER reduces context length and removes distracting information for downstream LMs.

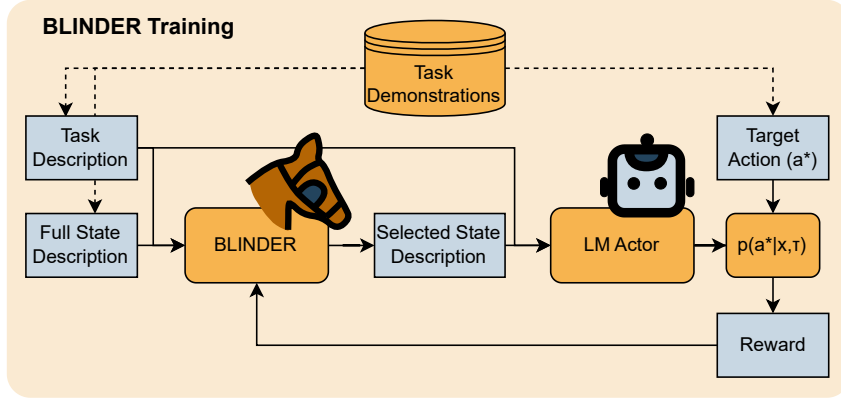


Figure 2: BLINDER is trained to produce minimal descriptions that maximize target action probability.

In this work, we develop a method for using a small finetuned LM (780M parameters) as a value function to select concise task-conditioned state descriptions from a set of provided state features (Figure 1). We call our method **BLINDER*** (**B**rief **L**anguage **I**nputs for **D**ecision-making **R**esponses). To learn concise inputs, BLINDER finds the minimal set of state features that maximizes task performance as predicted by a value function. We approximate task performance using an LM actor’s alignment with several task demonstrations (~ 5). Given a task demonstration and an LM actor, we assign rewards to state descriptions using the likelihood of target actions from the LM actor (Figure 2). We use these rewards to finetune an LM with a value head and, at inference time, sample minimal state descriptions that maximize the learned value function.

We evaluate our method on the grid-based video game environment NetHack (Küttler et al., 2020) and a real world robotic item arrangement task. BLINDER decreases the size of LM actor inputs by 87% and 99% while improving LM actor success rate by 158% and 54% on each task respectively. We also explore the trade off between summary quality and the size of a summarization model by comparing BLINDER to zeroshot summaries from pretrained LMs. Finally, we show that BLINDER’s learned state descriptions are intuitive enough to generalize across LM actors, allowing us to train with an open-source LM actor and transfer to a larger black-box actor. Overall, our research demonstrates that small finetuned LMs are well equipped to produce concise and helpful inputs, often beating larger zeroshot summarization models in both metrics.

*<https://kolbytn.github.io/blinder/>

2 BLINDER

We explore learning to select a concise state description from a set of state features given a task.

Algorithm 1 State Description Selection

Require: S, V_θ, τ
 $X \leftarrow \emptyset$
while $|S| > 0$ and $\max_{s \in S} (V_\theta(X + s, \tau)) > V_\theta(X, \tau)$
do
 $s \leftarrow \operatorname{argmax}_{s \in S} (V_\theta(X + s, \tau))$
 $X \leftarrow X + s$
 $S \leftarrow S - s$
end while
return X

Algorithm 1 defines our policy for selecting state descriptions using BLINDER, where S is the set of all available state features, τ is the task description, and V_θ is a state description value function. The resulting state description X is a minimal state description that maximizes the value estimate $V_\theta(X, \tau)$. Note that the worst case time complexity of Algorithm 1 is $O(|S|^2)$. However, in our experiments, the selection process always ended after 3-5 state features were selected.

We train V_θ to estimate downstream performance by leveraging a small set of trajectory demonstrations (~ 5), composed of state features, task descriptions, and target actions $D = \{(S, \tau, a^*), \dots\}$. For each trajectory step, we sample state descriptions X for training and define a sparse reward function R_{LM} for assigning rewards for X ,

$$R_{LM}(X, \tau, a^*) = LM(a^*|X, \tau). \quad (1)$$

R_{LM} maximizes the likelihood that X elicits the target action a^* from a pretrained LM actor. Note that, although R_{LM} is defined for a specific LM actor, we find that BLINDER selects intuitive state features and generalizes well to other LM actors.

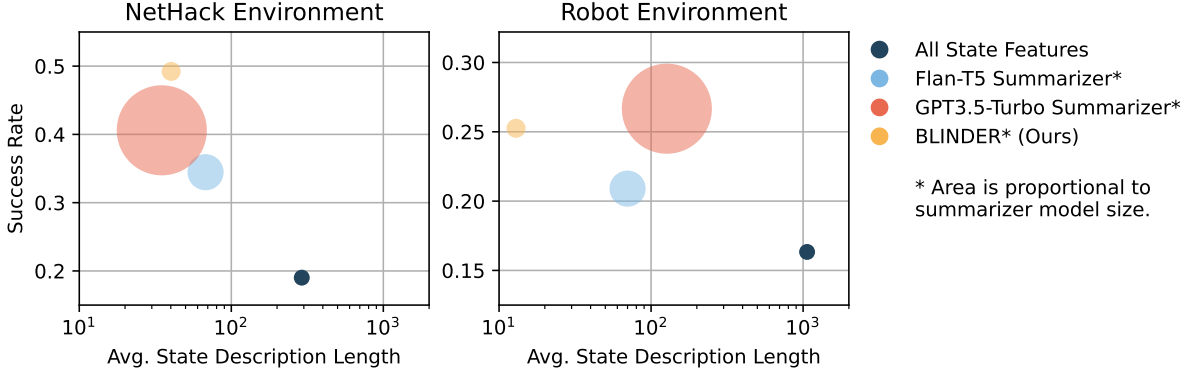


Figure 3: Success rate vs. average state description length vs. summarizer model size on NetHack and Robot environments. For visualization purposes, we approximate gpt3.5-turbo as 25x the size of BLINDER. Better performance and lower compute is represented by smaller circles up and to the left.

Finally, we define the loss function of a single state description

$$\mathcal{L}(X, S, \tau, a^*) = \sum_{t=0}^{|X|} \left(V_{\theta}(X_{:t}, \tau) - \gamma^{|X|-t} R_{LM}(X, \tau, a^*) \right)^2 \quad (2)$$

and the overall loss function for V_{θ}

$$\mathcal{L}_{V_{\theta}} = \mathbb{E}_{\substack{S, \tau, a^* \sim D \\ X \sim \pi | S, \tau}} \left[\mathcal{L}(X, S, \tau, a^*) + \phi \right]. \quad (3)$$

where γ is a discount factor and ϕ is a Kullback-Leibler penalty for normalizing V_{θ} , common when finetuning LMs with RL (Stiennon et al., 2020b; Leblond et al., 2021). At inference time, X is always sampled using Algorithm 1. However, when training V_{θ} , we found sampling from a random policy to be sufficient and more efficient.

3 Experimental Results

3.1 Setup

We train BLINDER using NetHack tasks from the Minihack environment zoo (Samvelyan et al., 2021) with the recommended natural language wrapper (Goodger et al., 2023). We then evaluate on more complex variations of the Minihack zoo tasks (Appendix C).

We also train BLINDER on a real-world robotic object arrangement task. In this task, several objects must be rearranged on a table within a 2x5 grid in order from left to right. Distractor objects

are also placed in this grid to complicate the state space. We evaluate on unseen items (Appendix D).

Our experiments use three baselines to evaluate BLINDER. First, we compare with the full set of state features that includes all S . We also compare against zeroshot summarization from a flan-t5-x1 and gpt3.5-turbo model.

BLINDER finetunes a 780 million parameter flan-t5-large model with a value head for V_{θ} . It is rewarded using a 3 billion flan-t5-x1 actor (Appendix B.1). We also test BLINDER with a larger gpt3.5-turbo actor (Appendix B.2) to demonstrate its ability to generalize to large actors and reduce compute costs.

3.2 Analysis

Figure 3 compares success rate to the state description length for each task domain. Success rates are averaged over a set of held out test tasks in each domain, and state description lengths are the average number of tokens in summaries generated by each summarizer model and BLINDER. The number of tokens in the state descriptions directly increases the compute costs of the downstream actor.

BLINDER consistently improves state descriptions composed of all state features and samples shorter and better summaries than those generated by the flan-t5-x1 summarizer. BLINDER also remains competitive with summaries generated by gpt3.5-turbo despite the latter being significantly larger. It is also worth noting that BLINDER is the only method we test that never requires having the entire state in context at once since it iteratively evaluates features. This is helpful in situations in which the exhaustive state description would exceed LM context limits.

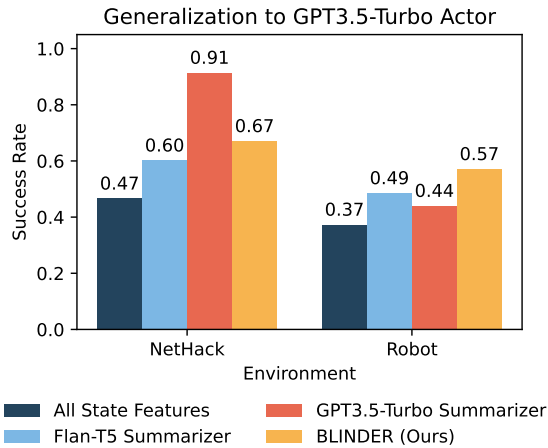


Figure 4: BLINDER generalizes to other LM actors. Description length and model size are equal to Figure 3.

BLINDER also creates intuitive state descriptions that generalize between LM actors. Although BLINDER is trained to optimize inputs for the `flan-t5-xl` actor, its state descriptions are intuitive enough to generalize well to the much larger `gpt3.5-turbo` actor achieving better success rate than the LM actor it was trained with. In this setup, the `gpt3.5-turbo` summarizer outperforms BLINDER on the NetHack task. However, with many more parameters, `gpt3.5-turbo` forfeits the gains in compute efficiency that BLINDER reaches. Also, the `gpt3.5-turbo` summarizer’s performance gains are inconsistent. It still struggles on the Robot arrangement task likely because its state descriptions remain lengthy compared to those of BLINDER as seen in Figure 3.

4 Related Work

4.1 LMs for Planning

LMs have recently become popular for planning and high-level decision making in robotic tasks (Ichter et al., 2022; Huang et al., 2022a, 2023; Vemprala et al., 2023) and other sequential decision making benchmarks (Nottingham et al., 2023; Kim et al., 2023; Liu et al., 2023a; Liang et al., 2023). While learned techniques exist for grounding LM actors in the state (Ichter et al., 2022), the most straightforward way to ground LMs is to include state features in the input (Huang et al., 2022b; Liang et al., 2022; Singh et al., 2022; Skreta et al., 2023; Zhao et al., 2023; Lin et al., 2023; Wake et al., 2023). These state descriptions often contain needless information, and more hand-engineered state descriptions do not generalize between tasks.

4.2 Learned Inputs for LMs

Despite increase in supported context lengths, LMs struggle to make use of very long inputs (Liu et al., 2023b; Qin et al., 2023). Retrieval methods that rely on similarity metrics are commonly used (Nogueira and Cho, 2019; Karpukhin et al., 2020; Lewis et al., 2020) but not helpful for selecting input features that may be dissimilar to the query text. Some retrieval methods for documents (Wang et al., 2018) or fewshot examples (Rubin et al., 2022; Wang et al., 2023; Gupta et al., 2023) use learned values. However, these approaches do not consider when to stop adding examples with the objective of concise inputs. An extensive range of studies investigate learning instructions or prompt formats for improving LM performance on a dataset in discrete (Shin et al., 2020; Gao et al., 2021; Shi et al., 2022; Deng et al., 2022; Fernando et al., 2023; ?) or continuous (Qin and Eisner, 2021; Liu et al., 2021; Lester et al., 2021; Zhong et al., 2021) input space. However, the objective of previous work is only to increase performance by modifying instructions and ignores the problem of conciseness. Recent methods have explored the idea of compressing input tokens for shorter context length (Mu et al., 2023; Chevalier et al., 2023) but require finetuning the downstream LM.

5 Conclusion

In this work, we explore using small finetuned LMs to select concise state descriptions that reduce compute and improve performance. Rather than restrict prompt tuning to task examples or instructions as in previous work, we propose modifying the entire input space. Our results indicate that small finetuned LMs are well equipped to process entire inputs for larger more powerful LMs to use on downstream tasks. Zeroshot summarization from large LMs is another possible approach to decreasing the length of input for a downstream LM. Zeroshot summaries have the benefit of not requiring a user to finetune a model, but they do not perform as consistently and generally require larger models than their finetuned counterparts such as BLINDER. We find that BLINDER makes a significant difference in both downstream performance and overall inference costs. We hope to encourage continued research into learning more efficient LM contexts through methods such as pruning, summarization, and retrieval.

6 Acknowledgment

This material is sponsored in part by the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research and the NSF CAREER award number IIS-2046873 and NSF CNS-1925741 fund.

Limitations

Our approach assumes a set of order invariant and independent state features. While this is a simple assumption for embodied decision making, where state features are often already a set of entities and features, this is not true for most natural language processing tasks. Future work could go into addressing a generalized version of our approach that supports any text input.

Additionally, BLINDER requires a small set of labeled trajectories to approximate task performance. This is a simpler requirement than hand engineering state descriptions, but it still requires some work from humans. It should be possible to instead estimate task performance by doing several task rollouts with the current summarizer and policy, but this requires significantly more compute than our approach.

References

- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. [Scaling instruction-finetuned language models](#).
- Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yi-han Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. [RLPrompt: Optimizing discrete text prompts with reinforcement learning](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. [Making pre-trained language models better few-shot learners](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.
- Nikolaj Goodger, Peter Vamplew, Cameron Foale, and Richard Dazeley. 2023. [A nethack learning environment language wrapper for autonomous agents](#). *Journal of Open Research Software*, 11.
- Agrim Gupta, Piotr Dollar, and Ross Girshick. 2019. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364.
- Shivanshu Gupta, Clemens Rosenbaum, and Ethan R Elenberg. 2023. Gistscore: Learning better representations for in-context example selection with gist bottlenecks. *arXiv preprint arXiv:2311.09606*.
- Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.
- Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.
- Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. 2023. Grounded decoding: Guiding text generation with grounded models for robot control. *arXiv preprint arXiv:2303.00855*.
- Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022b. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*.
- Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu,

- Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. 2022. [Do as i can, not as i say: Grounding language in robotic affordances](#). In *6th Annual Conference on Robot Learning*.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.
- Charles C Kemp, Aaron Edsinger, Henry M Clever, and Blaine Matulevich. 2022. The design of stretch: A compact, lightweight mobile manipulator for indoor human environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3150–3157. IEEE.
- Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*.
- Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. 2020. [The nethack learning environment](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 7671–7684. Curran Associates, Inc.
- Rémi Leblond, Jean-Baptiste Alayrac, Laurent Sifre, Miruna Pislariu, Lespiau Jean-Baptiste, Ioannis Antonoglou, Karen Simonyan, and Oriol Vinyals. 2021. Machine translation decoding beyond beam search. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8410–8434.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Pete Florence, Andy Zeng, et al. 2022. Code as policies: Language model programs for embodied control. In *Workshop on Language and Robotics at CoRL 2022*.
- Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2023. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*.
- Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. 2023. Text2motion: From natural language instructions to feasible plans. *arXiv preprint arXiv:2303.12153*.
- Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.
- Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. [Gpt understands, too](#).
- Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, page 3721. Association for Computational Linguistics.
- Pierre Manceron. 2022. [Ikpy](#). If you use this software, please cite it using the metadata from this file.
- Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. [Learning to compress prompts with gist tokens](#).
- Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.
- Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. 2023. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. *arXiv preprint arXiv:2301.12050*.
- Guanghui Qin and Jason Eisner. 2021. [Learning how to ask: Querying LMs with mixtures of soft prompts](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.
- Guanghui Qin, Yukun Feng, and Benjamin Van Durme. 2023. The nlp task effectiveness of long-range transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3756–3772.
- Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. [Learning to retrieve prompts for in-context learning](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

- Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. 2021. Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.
- Weijia Shi, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. 2022. Toward human readable prompt tuning: Kubrick’s the shining is a good movie, and a good prompt too? *arXiv preprint arXiv:2212.10539*.
- Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.
- Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2022. Prog-prompt: Generating situated robot task plans using large language models. In *Second Workshop on Language and Reinforcement Learning*.
- Marta Skreta, Naruki Yoshikawa, Sebastian Arellano-Rubach, Zhi Ji, Lasse Bjørn Kristensen, Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. 2023. Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting. *arXiv preprint arXiv:2303.14100*.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020a. [Learning to summarize with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020b. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.
- Sai Vemprala, Rogerio Bonatti, Arthur Buckner, and Ashish Kapoor. 2023. Chatgpt for robotics: Design principles and model abilities. *Microsoft*.
- Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. 2023. Chatgpt empowered long-step robot control in various environments: A case application. *arXiv preprint arXiv:2304.03893*.
- Liang Wang, Nan Yang, and Furu Wei. 2023. Learning to retrieve in-context examples for large language models. *arXiv preprint arXiv:2307.07164*.
- Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesaro, Bowen Zhou, and Jing Jiang. 2018. R 3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. <https://github.com/facebookresearch/detectron2>.
- Xufeng Zhao, Mengdi Li, Cornelius Weber, Muhammad Burhan Hafez, and Stefan Wermter. 2023. Chat with the environment: Interactive multimodal perception using large language models. *arXiv preprint arXiv:2303.08268*.
- Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. [Factual probing is \[MASK\]: Learning vs. learning to recall](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

Appendix

A Method Details

Hyperparameter	Value
γ	1
lr	1e-6
batch size	4
KL regularization coefficient	1
max state description length (nethack)	10
max state description length (robot)	5

Table 1: BLINDER training hyperparameters

A.1 BLINDER

Table 1 shows hyperparameters used during BLINDER training. We finetune a flan-t5-large model with a value head after the final hidden state in the decoder. We prompt the model with the following instructions to take advantage of flan-T5’s instruction finetuning:

Describe the relevant information from the game state for the current task. Your current task is to [task description].

A.2 Baselines

We prompt flan-t5-xl and gpt3.5-turbo to zeroshot summarize the state features to compare with BLINDER. Although these methods have some success, they require larger models than our method. We use the following prompt for summarization:

Prune the sentences from the original text such that only information relevant to your current task remains. Only output sentences that appear in the original text. Your current task is to [task description].

B Actor Details

B.1 T5 Actor

Our **flan-T5 actor** is a three billion parameter flan-t5-xl model (Chung et al., 2022). We sample actions from this actor by computing the geometric mean of the logits for each environment action, taking the softmax, and sampling from the resulting distribution. This is the model that BLINDER uses for training.

We prompt a pretrained flan-t5-xl model with the below prompts for use as an LM actor:

NetHack Domain:

You are playing the rogue-like game NetHack. Your task is to [task description]. You can move north, south, east, west, northeast, southeast, southwest, or northwest. You can attack monsters adjacent to you, pick up items under you, zap wands, eat food, wear armor, use keys, drink potions, and put on rings. [state description]

You choose to:

Robot Domain:

You are controlling a helpful household robot. Your task is to [task description]. You can move items from their current positions to empty positions indicated by their corresponding letter. [state description]

You choose to:

At each task step, we compute the likelihood of the admissible actions, normalize the probabilities using the softmax function, and then sample an action to execute in the environment.

B.2 GPT Actor

Our **GPT3.5 Turbo actor** uses `gpt3.5-turbo-0301`[†] to generate an action from a list of admissible actions given a task and state description and six fewshot examples. We use this model to evaluate BLINDER’s ability to generalize between LM actors.

`gpt3.5-turbo-0301` is prompted with the below system messages:

NetHack Domain:

You are playing the rogue-like game NetHack. You can move north, south, east, west, northeast, southeast, southwest, or northwest. You can attack monsters adjacent to you, pick up items under you, zap wands, eat food, wear armor, use keys, drink potions, and put on rings.

Robot Domain:

You are controlling a helpful household robot. You can move items from their current positions to empty positions indicated by their cooresponding letter.

We provide fewshot examples and prompt the actor with the below prompt:

Your task is to [task description].

Game Description:

[state description]

Choose the best action:

[list of admissible actions]

The model generates an admissible action that is executed in the environment.

[†]<https://platform.openai.com/docs/models/gpt-3-5>



Figure 5: Example lava cross task.

Actor	Summarizer	NetHack Task					avg
		paer	shoes	potion	ring	boots	
Flan-T5-xl	None	.1585	.1585	.2575	.1684	.2080	.1902
Flan-T5-xl	Flan-T5-xl	.4753	.4951	.3268	.2773	.1486	.3446
Flan-T5-xl	GPT3.5-Turbo	.4900	.6700	.3400	.1700	.3600	.4060
Flan-T5-xl	BLINDER	.4951	.3961	.5700	.4753	.5248	.4923
GPT3.5-Turbo	None	.9785	.8315	.4060	.3070	.4852	.6016
GPT3.5-Turbo	Flan-T5-xl	.7113	.7576	.1684	.2872	.4060	.4661
GPT3.5-Turbo	GPT3.5-Turbo	.9100	.8900	.9900	.8800	.9000	.9140
GPT3.5-Turbo	BLINDER	.9139	.9348	.4747	.6436	.3862	.6706

Table 2: All success rates on NetHack test tasks.

C NetHack Details

We evaluate BLINDER on NetHack tasks using the Minihack library (Samvelyan et al., 2021). NetHack, recently proposed as a testbed for AI research (Küttler et al., 2020), is a grid-based dungeon crawler with complex dynamics and large observation and action spaces.

We obtain S for NetHack from the natural language wrapper recommended by Küttler et al. (2020)[‡]. The resulting sets of state features contain an average of 35 state features.

We select a set of five training tasks from the Minihack environment zoo: Room-Monster-5x5, Eat, Wear, LavaCross-Levitate-Potion-Inv, and LavaCross-Levitate-Ring-Inv. We use just 25 expert trajectories for training with a total of 148 pairs of states and expert actions that required under an hour for a human annotator to collect.

We design five difficult custom test tasks to evaluate the performance of BLINDER. Two tasks, Eat paer and Wear shoes, are variants of the Eat and Wear training tasks but with different target items, larger rooms, monsters, and distractor items. We also define three Lava cross test tasks. Unlike the training variants of this task, the item needed to cross the lava does not start in the player inventory, necessitating improved multi-step planning from the LM actor. The boots variant of the lava cross task was not seen during training. See Table 6 for an example trajectory.

[‡]<https://github.com/ngoodger/nle-language-wrapper>

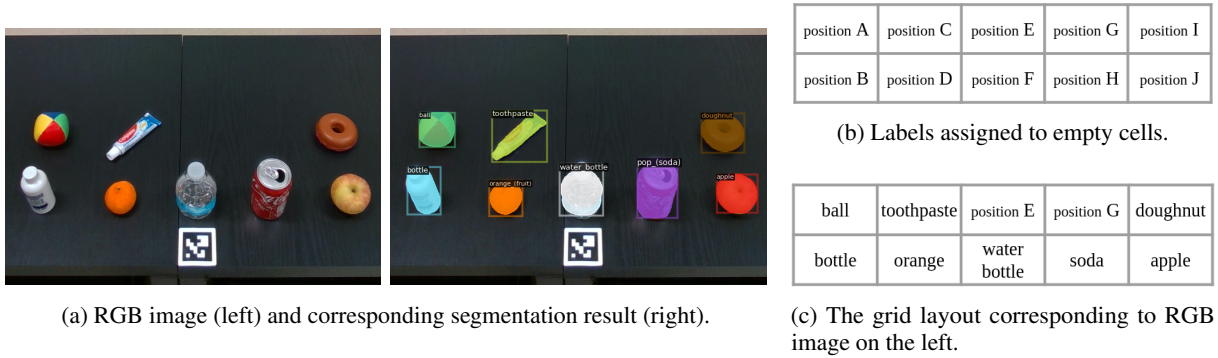


Figure 6

Actor	Summarizer	Robot Arrangement Size			
		2	3	4	avg
Flan-T5-xl	None	.3500	.1100	.0300	.1633
Flan-T5-xl	Flan-T5-xl	.3500	.2080	.0690	.2090
Flan-T5-xl	GPT3.5-Turbo	.4800	.2600	.0600	.2667
Flan-T5-xl	BLINDER	.5600	.1480	.0496	.2525
GPT3.5-Turbo	None	.8812	.4853	.0892	.4852
GPT3.5-Turbo	Flan-T5-xl	.7900	.2800	.0490	.3730
GPT3.5-Turbo	GPT3.5-Turbo	.8200	.4600	.0400	.4400
GPT3.5-Turbo	BLINDER	.9406	.6634	.1090	.5710

Table 3: All success rates on test robot arrangement tasks.

D Robot Task Details

D.1 Robot Details

For the robotics experiment, we use the Stretch RE2 (Kemp et al., 2022) from Hello Robot Inc. Stretch is a lightweight, low-cost mobile manipulator equipped with a variety of sensors, including an RGB-D camera and a 2D LiDAR (see Figure 7). The Stretch uses an Intel RealSense D435i camera to collect RGB and depth images of the table and we use both the onboard RP-LiDAR A1 and an added HTC Vive motion tracker for position estimates.

D.2 Environment Details

The goal of the robot planning task is to rearrange several objects on a table in a target order from left to right. Objects may each occupy and be placed in predefined locations in a 2-row by 5-column grid on the table. Target object arrangements are defined by the horizontal order of each item, and an object’s final row location does not affect task success. A trial is successful if the robot arranges the objects in a specified order in 10 actions or less.

To investigate the generalization ability of BLINDER, we test on held-out items not seen during training. Additionally, while agents are tasked with arranging only two or three items during training, they are evaluated on arranging two, three, and four items at test time.

D.2.1 Observation space

Using LMs as high-level planners requires natural language input, however object locations are perceived by the Stretch robot with

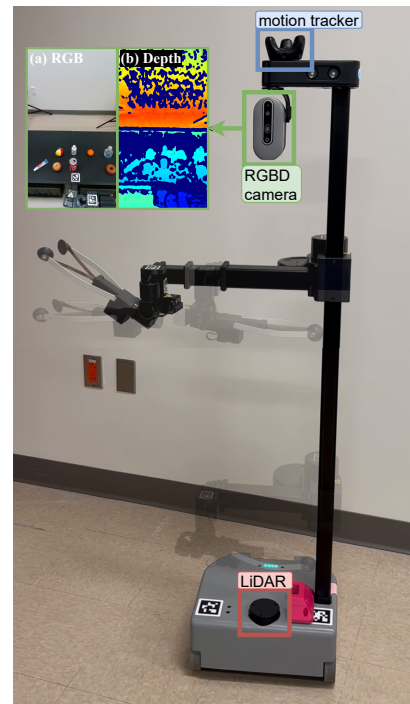


Figure 7: Stretch RE2.

camera inputs. We apply a pipeline to parse image observations to a canonical state representation and then to natural language state descriptions.

Given an RGB image of the table, we detect table objects with an off-the-shelf semantic segmentation mask and determine their 3D location using a corresponding depth image. The locations of grid cells are implicitly defined relative to the table. The canonical state we extract describes which objects are in which grid cells. For segmentation, we use a Mask R-CNN (He et al., 2017) model with a ResNet-101-FPN backbone pre-trained on LVIS (Gupta et al., 2019) using the Detectron2 library (Wu et al., 2019). Figure 6a shows an example of an RGB image and its corresponding segmentation result. Figure 6c shows the parsed grid state corresponding to the image in Figure 6a.

Next, we construct a natural language state description from the current grid layout by listing spatial relationships between each grid cell and every other grid cell. Spatial relationships include left, right, behind, and beyond. For grid cells that are populated, the residing object labels are used as identifiers for the grid cells. Table 5 shows an example of a grid state being converted to a full natural language state description.

D.2.2 Action space

The high-level action defined for object arrangement follows the format “*move object_name to empty_position_name*”. In our experiments, the Stretch robot achieves this by using 3D object coordinates obtained from segmented RGB-D camera images in combination with IKPy (Manceron, 2022), a python library for inverse kinematics.

NetHack

Full State Description



“You have a +1 club (weapon in hand). You have a +2 sling (alternate weapon; not wielded). You have 16 uncursed flint stones (in quiver pouch). You have 25 uncursed rocks. You have an uncursed +0 leather armor (being worn). Strength: 22/19. Dexterity: 13. Constitution: 15. Intelligence: 9. Wisdom: 10. Charisma: 6. Depth: 1. Gold: 0. HP: 16/16. Energy: 2/2. AC: 8. XP: 1/0. Time: 1. Position: 3619. Hunger: Not Hungry. Monster Level: 0. Encumbrance: Unencumbered. Dungeon Number: 0. Level Number: 1. Score: 0. Alignment: Neutral. Condition: None. You see a vertical wall far east. You see a stairs down near east southeast. You see a horizontal wall near southeast and south. You see a lava near south southeast. You see a southwest corner near southwest. You see a vertical wall near west. You see a horizontal wall very near north, northeast, and northwest. You see a lava very near east northeast, east, east southeast, and southeast. You see a effervescent potion very near south southwest. Hello Agent, welcome to NetHack! You are a neutral human Caveman”

Table 4: The “Lava Cross: Potion” NetHack task observation and full state description.

Grid

Full State Description

ball	toothpaste	position E	position G	doughnut
bottle	orange	water bottle	soda	apple

“position E is behind the water bottle. position E is to the left of and behind the apple. position E is to the left of and behind the soda. position E is to the left of position G. position E is to the left of the doughnut. position E is to the right of and behind the bottle. position E is to the right of and behind the orange. position E is to the right of the ball. position E is to the right of the toothpaste. position G is behind the soda. position G is to the left of and behind the apple. position G is to the left of the doughnut. position G is to the right of and behind the bottle. position G is to the right of and behind the orange. position G is to the right of and behind the water bottle. position G is to the right of position E. position G is to the right of the ball. position G is to the right of the toothpaste. the apple is beyond the doughnut. the apple is to the right of and beyond position E. the apple is to the right of and beyond position G. the apple is to the right of and beyond the ball. the apple is to the right of and beyond the toothpaste. the apple is to the right of the bottle. the apple is to the right of the orange. the apple is to the right of the soda. the apple is to the right of the water bottle. the ball is behind the bottle. the ball is to the left of and behind the apple. the ball is to the left of and behind the orange. the ball is to the left of and behind the soda. the ball is to the left of and behind the water bottle. the ball is to the left of position E. the ball is to the left of position G. the ball is to the left of the doughnut. the ball is to the left of the toothpaste. the bottle is beyond the ball. the bottle is to the left of and beyond position E. the bottle is to the left of and beyond position G. the bottle is to the left of and beyond the doughnut. the bottle is to the left of and beyond the toothpaste. the bottle is to the left of the apple. the bottle is to the left of the orange. the bottle is to the left of the soda. the bottle is to the left of the water bottle. the doughnut is behind the apple. the doughnut is to the right of and behind the bottle. the doughnut is to the right of and behind the orange. the doughnut is to the right of and behind the soda. the doughnut is to the right of and behind the water bottle. the doughnut is to the right of position E. the doughnut is to the right of position G. the doughnut is to the right of the ball. the doughnut is to the right of the toothpaste. the orange is beyond the toothpaste. the orange is to the left of and beyond position E. the orange is to the left of and beyond position G. the orange is to the left of and beyond the doughnut. the orange is to the left of the apple. the orange is to the left of the soda. the orange is to the left of the water bottle. the orange is to the right of and beyond the ball. the orange is to the right of the bottle. the soda is beyond position G. the soda is to the left of and beyond the doughnut. the soda is to the left of the apple. the soda is to the right of and beyond position E. the soda is to the right of and beyond the ball. the soda is to the right of and beyond the toothpaste. the soda is to the right of the bottle. the soda is to the right of the orange. the soda is to the right of the water bottle. the toothpaste is behind the orange. the toothpaste is to the left of and behind the apple. the toothpaste is to the left of and behind the soda. the toothpaste is to the left of and behind the water bottle. the toothpaste is to the left of position E. the toothpaste is to the left of position G. the toothpaste is to the left of the doughnut. the toothpaste is to the right of and behind the bottle. the toothpaste is to the right of the ball. the water bottle is beyond position E. the water bottle is to the left of and beyond position G. the water bottle is to the left of and beyond the doughnut. the water bottle is to the left of the apple. the water bottle is to the left of the soda. the water bottle is to the right of and beyond the ball. the water bottle is to the right of and beyond the toothpaste. the water bottle is to the right of the bottle. the water bottle is to the right of the orange”

Table 5: Robot arrangement task observation and full state description.




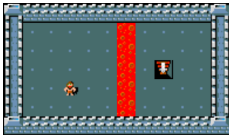

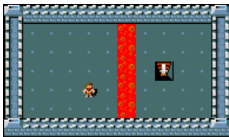
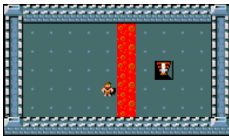
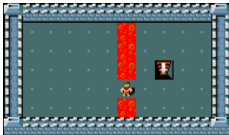

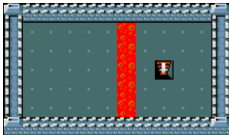
Time	Image	Selected State Description	Action
t=0		You see a effervescent potion very near south southwest. You see a stairs down near east southeast. You see a lava near south southeast. You have a +2 sling (alternate weapon; not wielded).	move southwest
t=1		You see a effervescent potion very near south southwest. You see a stairs down near east southeast. You see a lava near south southeast. You have a +2 sling (You see a effervescent potion adjacent south. You have a +2 sling (alternate weapon; not wielded). Score: 0. You have an uncursed +0 leather armor (being worn). You have 16 uncursed flint stones (in quiver pouch).	move south
t=2		You see here an effervescent potion. You have a +2 sling (alternate weapon; not wielded). Score: 0. Depth: 1.	pick up an effervescent potion
t=3		f - an effervescent potion. You have a +2 sling (alternate weapon; not wielded). Score: 0. Encumbrance: Unencumbered.	drink an effervescent potion
t=4		Dexterity: 13. You have 16 uncursed flint stones (in quiver pouch). You have 25 uncursed rocks.	move east
t=5		Dexterity: 13. You have 16 uncursed flint stones (in quiver pouch). You have 25 uncursed rocks.	move east
t=6		Dexterity: 13. You have 16 uncursed flint stones (in quiver pouch). You have 25 uncursed rocks.	move east
t=7		Dexterity: 13. You have 16 uncursed flint stones (in quiver pouch). You have 25 uncursed rocks.	move east
t=8		You see a stairs down adjacent northeast. You have 16 uncursed flint stones (in quiver pouch). You have 25 uncursed rocks.	move northeast
t=9		-	-

Table 6: Here, our fewshot actor is prompted with “Your task is to pick up and drink the potion and navigate to the stairs down.”. The table shows a NetHack visualization alongside selected state descriptions and the action selected by the LM actor.




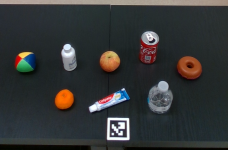
Time	Image	Grid	Selected State Description	Action										
t=0		<table border="1"> <tr> <td>position A</td> <td>bottle</td> <td>apple</td> <td>doughnut</td> <td>position I</td> </tr> <tr> <td>soda</td> <td>orange</td> <td>toothpaste</td> <td>water bottle</td> <td>ball</td> </tr> </table>	position A	bottle	apple	doughnut	position I	soda	orange	toothpaste	water bottle	ball	position A is to the left of the doughnut. position A is to the left of and behind the ball.	move the ball to position A.
position A	bottle	apple	doughnut	position I										
soda	orange	toothpaste	water bottle	ball										
t=1		<table border="1"> <tr> <td>ball</td> <td>bottle</td> <td>apple</td> <td>doughnut</td> <td>position I</td> </tr> <tr> <td>soda</td> <td>orange</td> <td>toothpaste</td> <td>water bottle</td> <td>position J</td> </tr> </table>	ball	bottle	apple	doughnut	position I	soda	orange	toothpaste	water bottle	position J	position I is behind position J. the doughnut is to the left of position I.	move the doughnut to position I.
ball	bottle	apple	doughnut	position I										
soda	orange	toothpaste	water bottle	position J										
t=2		<table border="1"> <tr> <td>ball</td> <td>bottle</td> <td>apple</td> <td>position G</td> <td>doughnut</td> </tr> <tr> <td>soda</td> <td>orange</td> <td>toothpaste</td> <td>water bottle</td> <td>position J</td> </tr> </table>	ball	bottle	apple	position G	doughnut	soda	orange	toothpaste	water bottle	position J	position G is to the left of the doughnut. position G is to the right of the ball.	move the soda to position G.
ball	bottle	apple	position G	doughnut										
soda	orange	toothpaste	water bottle	position J										
t=3		<table border="1"> <tr> <td>ball</td> <td>bottle</td> <td>apple</td> <td>soda</td> <td>doughnut</td> </tr> <tr> <td>position B</td> <td>orange</td> <td>toothpaste</td> <td>water bottle</td> <td>position J</td> </tr> </table>	ball	bottle	apple	soda	doughnut	position B	orange	toothpaste	water bottle	position J	-	-
ball	bottle	apple	soda	doughnut										
position B	orange	toothpaste	water bottle	position J										

Table 7: Successful trajectory for the task “Arrange the objects in the order: **ball, soda, doughnut.**” The items relevant to the task are highlighted – **green color** denotes correct alignment and **red** means incorrect alignment.



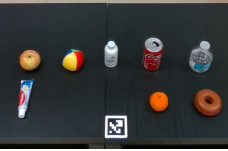
Time	Image	Grid	Selected State Description	Action										
t=0		<table border="1"> <tr> <td>apple</td> <td>position C</td> <td>bottle</td> <td>soda</td> <td>water bottle</td> </tr> <tr> <td>ball</td> <td>toothpaste</td> <td>position F</td> <td>orange</td> <td>doughnut</td> </tr> </table>	apple	position C	bottle	soda	water bottle	ball	toothpaste	position F	orange	doughnut	the bottle is to the right of position C. position C is behind the toothpaste. position C is to the right of and behind the ball.	move the ball to position C.
apple	position C	bottle	soda	water bottle										
ball	toothpaste	position F	orange	doughnut										
t=1		<table border="1"> <tr> <td>apple</td> <td>ball</td> <td>bottle</td> <td>soda</td> <td>water bottle</td> </tr> <tr> <td>position B</td> <td>toothpaste</td> <td>position F</td> <td>orange</td> <td>doughnut</td> </tr> </table>	apple	ball	bottle	soda	water bottle	position B	toothpaste	position F	orange	doughnut	position B is to the left of the toothpaste. the ball is behind the toothpaste. position F is to the right of position B. the orange is to the right of position B. the soda is to the right of the ball.	move the toothpaste to position B.
apple	ball	bottle	soda	water bottle										
position B	toothpaste	position F	orange	doughnut										
t=2		<table border="1"> <tr> <td>apple</td> <td>ball</td> <td>bottle</td> <td>soda</td> <td>water bottle</td> </tr> <tr> <td>toothpaste</td> <td>position D</td> <td>position F</td> <td>orange</td> <td>doughnut</td> </tr> </table>	apple	ball	bottle	soda	water bottle	toothpaste	position D	position F	orange	doughnut	-	-
apple	ball	bottle	soda	water bottle										
toothpaste	position D	position F	orange	doughnut										

Table 8: Successful trajectory for the task “Arrange the objects in the order: **toothpaste, ball, bottle, soda.**” Notably, BLINDER can be generalized to arranging an unseen number of items.