PROMPTS AND PRE-TRAINED LANGUAGE MODELS FOR OFFLINE REINFORCEMENT LEARNING

Denis Tarasov, Vladislav Kurenkov & Sergey Kolesnikov[*]
Tinkoff
Moscow, Russia
{ext.dtarasov, v.kurenkov, s.s.kolesnikov}@tinkoff.ai

ABSTRACT

In this preliminary study, we introduce a simple way to leverage pre-trained language models in deep offline RL settings that are not naturally suited for textual representation. We propose using a state transformation into a human-readable text and a minimal fine-tuning of the pre-trained language model when training with deep offline RL algorithms. This approach shows consistent performance gains on the NeoRL MuJoCo datasets. Our experiments suggest that LM fine-tuning is crucial for good performance on robotics tasks. However, we also show that it is not necessary when working with finance environments in order to retain significant improvement in the final performance.

1 INTRODUCTION

The interest in applying natural language to reinforcement learning is rapidly growing. One of this sphere's underexplored directions is the incorporation of pre-trained language models into deep RL algorithms where the environments are not naturally suited for textual representation.

Recently, Reid et al. (2022) took a step in this direction by taking an earlier proposed Decision Transformer (Chen et al., 2021) and matching it with a pre-trained language model, demonstrating remarkable results on several deep offline RL benchmarks. However, this method is designed specifically for the Decision Transformer, and there is a plethora of other algorithms available.

In this work, we demonstrate that there is a simpler way to combine pre-trained language models (P-LMs) with deep offline RL algorithms for environments with vector-like observations. We propose using prompt engineering (Liu et al., 2021a) by presenting the vector state as a string, embedding it via P-LM, and then concatenating the resulting encoding with the original observation. This results in state-of-the-art performance on a subset of the NeoRL benchmark (Qin et al., 2021) across model-free algorithms.

2 BACKGROUND

2.1 OFFLINE RL

Reinforcement Learning (RL) is commonly defined by Markov Decision Process (MDP) with a tuple (S, A, P, r, γ), where S is a state space, A is an action space, $P(s'|s, a)$ is a transition function, $r(s, a)$ is a reward function, and γ is a discount factor. The goal of the agent is to obtain a policy $\pi(s, a)$ that maximizes $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right]$ by interacting with the MDP.

Offline RL is different from RL, as its agent cannot interact with the environment and the policy must be obtained by learning from the dataset D collected with some behavior policy π_β or a set of such policies. The main challenge of offline RL is that D usually does not contain all possible transactions while the agent does not have any opportunity to explore. This is one of the reasons why this problem is considered challenging, as the agent cannot improve on incorrect estimations of states or actions if they are not presented in D (Levine et al., 2020).

[*]First two authors contributed equally.

2.2 Language Models in RL

There is a considerable amount of recent work that employs pre-trained LMs for solving RL tasks. For example, Majumdar et al. (2020) use pre-trained ViLBERT (Lu et al., 2019) to solve vision-and-language navigation tasks, and Yao et al. (2020) show that fine-tuning GPT-2 (Radford et al., 2019) for action generation boosts performance in text-based games. In another work, Singh et al. (2021) fine-tune DistilBERT (Sanh et al., 2019) to encode states and actions in text-based games, and this approach obtains higher scores comparing to those that do not use pre-trained LMs. These studies show that knowledge stored in pre-trained LMs might be useful for solving RL tasks with natural language. For a more comprehensive overview, see Luketina et al. (2019); Madureira & Schlangen (2020).

While the aforementioned works focus on problems involving natural language, there is a recent interest in incorporating pre-trained LMs for problems not defined via natural language by default. In particular, Li et al. (2022) demonstrates that utilizing pre-trained LM for encoding both the goal, history, and parts of the observation for imitation learning leads to improved systematic generalization.

To the best of our knowledge, Reid et al. (2022) is the only existing work in offline reinforcement learning that adapts a pre-trained LM model for text-free RL tasks. The authors propose using a pre-trained LM to improve the convergence speed and the final performance of the Decision Transformer (Chen et al., 2021) by making the reward, action, and state encodings closer to the language model embeddings.

3 Proposed Approach

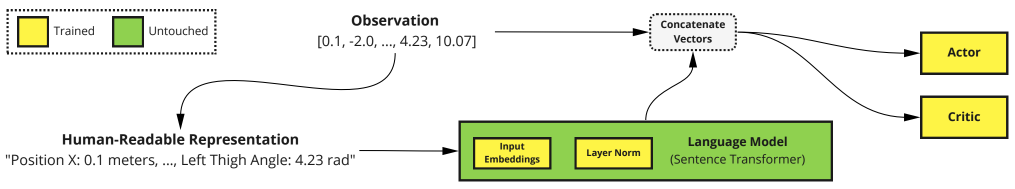


Figure 1: The proposed scheme for applying pre-trained language models in deep offline reinforcement learning. Note that the practitioner should only provide human-readable descriptions of the state dimensions. This scheme can be applied to most deep offline RL algorithms.

Inspired by the recent success in informing deep RL algorithms with pre-trained language models, we propose an alternative approach to leverage them in environments that are not obviously suited for the textual representation, similar to Reid et al. (2022); Li et al. (2022). Here, we suggest a method based on prompt-engineering (Liu et al., 2021a) depicted in Figure 1: (1) re-write an observation into a human-readable textual representation (2) encode it using a pre-trained language model, and (3) concatenate the resulting embedding with the original observation.

The first step can be an arduous task, requiring semantic parsing and bringing more structure to the problem. To alleviate this, we propose using a simple transformation that only requires giving a human-readable description of the dimensions of the state vector. For example, proprioceptive inputs are common in deep RL literature, and each dimension can usually be mapped to ”Position X: 10.0 meters” or ”Left Thigh Angle: 0.4 rad”. So the procedure is as follows: given a vector state s, we apply an environment-specific transformation $f : S \rightarrow String$ that converts each dimension value into a string, prepends a human-readable label to it, and then concatenates all of these strings into one.

In general, while pre-trained language models seem to handle human-readable texts well, token embeddings provided by pre-trained language models may not transfer between different domains: e.g., the corpus used for training the LM does not contain a lot of text with numbers representing robot joints or velocities. Therefore, in addition to re-writing the observation, we also perform minimal fine-tuning of the LM when training an offline RL algorithm. This is done in a way similar to Lu et al. (2021): instead of tuning the entire model, we only tune the input embeddings and layer norm parameters.

The proposed approach can be applied to any off-the-shelf offline RL algorithm, given that the length of the textual representation is suitable for the utilized LM. Note that it is not necessary to parse the observation space for meaningful entities, as it is enough to simply provide the labels for each dimension.

4 Experimental Evaluation

Concrete Realization The proposed method allows for various choices of language models, observation encodings, and deep offline RL algorithms. We use MPNET (Song et al., 2020) trained with Sentence Transformer (Reimers & Gurevych, 2019) as a backbone for the LM. For the textual representation, we round the numbers to 5 decimal places to speed up LM inference and reduce the memory space used by the model. A batch size of 32 is employed in order for the model to fit on a single NVIDIA Tesla V100 GPU with 32 GB of memory. We rely on TD3+BC (Fujimoto & Gu, 2021) as a deep offline RL algorithm in our experiments.

Benchmarks We probe our approach using a subset of NeoRL benchmark (Qin et al., 2021), running the experiments on the MuJoCo tasks, and report the best performance after a hyperparameter search. Following the original evaluation protocol by Qin et al. (2021), we train each policy for $3 \cdot 10^5$ iterations. However, we evaluate the policies with only 100 episodes instead of 1000, as we observed no difference in the final results. We also run the vanilla TD3+BC algorithm, as its results are not reported in the original NeoRL paper.

Table 1: Results for a subset of NeoRL dataset. The proposed approach outperforms the base model in two out of three settings. LM-No-Tune corresponds to a version of the method, where the language model is fixed throughout the training.

Task	BC	CQL	PLAS	BCQ	TD3+BC	TD3+BC LM-Finetune	TD3+BC LM-No-Tune
Hopper-v3-L-99	515	527	527	545	660	**762**	654
Walker2d-v3-L-99	1749	2370	42	1407	2480	**2669**	2564
HalfCheetah-v3-L-99	3260	3792	3451	3363	**4171**	4084	4171

The results of our preliminary set of experiments are presented in Table 1. We observe significant performance gains on two out of three settings using the proposed method (TD3+BC, LM-Finetune), and a slight drop on the third one.

Can We Avoid Fine-Tuning? To assess the importance of fine-tuning the input embeddings, we also run an ablation reported in Table 1 (denoted as LM-No-Tune). We observe that improvement is possible even without fine-tuning, but only on one of the environments and with a smaller magnitude.

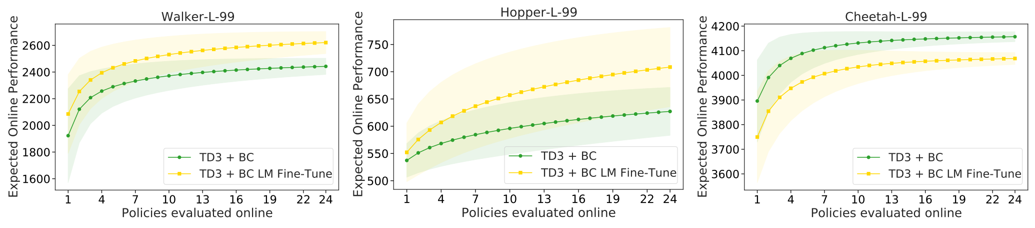
We hypothesize that this is due to a domain shift (Lu et al., 2021), i.e., it is possible that the training data for the LM did not contain much information about robot velocities and angles, and therefore some amount of fine-tuning is helpful. To test this further, we run the ablation in another environment – FinRL (Liu et al., 2021c). The observation space is also vector-based, but the values represent different financial indicators (e.g., price of a certain stock, an amount of a specific stock, etc.). We suppose that these indicators are more common than the proprioceptive inputs when training the LMs, and therefore a non-tuned LM should result in consistent improvements.

The results for various dataset levels and behavioral policies expertise can be found in Table 6. And indeed, we can see that no fine-tuning can be sufficient to achieve substantial improvements over the base model in most settings.

Table 2: Results for the FinRL subset of NeoRL benchmark. Tuning the language model may not be required to improve over the base model, where vector observations correspond to financial indicators.

Task	BC	CQL	PLAS	BCQ	TD3+BC	TD3+BC LM-No-Tune
FinRL-L-99	136	487	447	330	742	**1043**
FinRL-L-999	137	416	396	323	544	**627**
FinRL-M-99	355	700	388	376	808	**859**
FinRL-M-999	504	621	470	356	711	**804**
FinRL-H-99	252	671	464	426	**1024**	854
FinRL-H-999	270	444	495	330	879	**883**

Sensitivity to Online Evaluation Budget The preference between deep offline RL algorithms was noted to be dependent on the online evaluation budget available (Kurenkov & Kolesnikov, 2022). To analyze whether conclusions about the proposed method change under the varied budget, we also report the Expected Online Performance graphs in Figure 2. Our results indicate that, on environments where an improvement over the base model is possible, said improvement is achieved across all online budgets.



(a) Walker2d (b) Hopper (c) HalfCheetah

Figure 2: Expected Online Performance curves. When the proposed method outperforms the base model, it does so independent of the online evaluation budget.

5 Future Work and Limitations

Despite the encouraging results we observed in this preliminary study, we believe there are several important directions that should be explored and experiments to be done to substantiate the claims further:

- **Probe on a larger scale**: ablations with other deep offline RL algorithms, more environment and dataset sizes.
- **Systematic generalization**: while we performed testing on a common set of offline RL benchmarks, the most appealing side of the pre-trained language models is their ability to generalize. And while deep online RL algorithms were studied for systematic generalization (Jiang et al., 2021), their offline counterparts were not. We expect that the use of language models can substantially benefit deep offline RL algorithms in this aspect.
- **Explore different tuning approaches**: recent success in prompt-tuning (Liu et al., 2021b) for downstream NLP tasks suggests that the prompt engineering can be bypassed. We believe that exploring this method type in the context of deep RL can also be of interest.
- **Vision-based policies**: the proposed algorithm is not well-suited for vision-based policies, and studying how it can be adapted for this problem type is also a potential direction for future research.

REFERENCES

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/jiang21b.html`.

Vladislav Kurenkov and Sergey Kolesnikov. Showing your offline reinforcement learning work: Online evaluation budget matters, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021a.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks, 2021b.

Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. *ACM International Conference on AI in Finance (ICAIF)*, 2021c.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines, 2021.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

Brielen Madureira and David Schlangen. An overview of natural language state representation for reinforcement learning. *arXiv preprint arXiv:2007.09774*, 2020.

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

Rongjun Qin, Songyi Gao, Xingyuan Zhang, Zhen Xu, Shengkai Huang, Zewen Li, Weinan Zhang, and Yang Yu. Neorl: A near real-world benchmark for offline reinforcement learning. *arXiv preprint arXiv:2102.00714*, 2021.

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

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `http://arxiv.org/abs/1908.10084`.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Ishika Singh, Gargi Singh, and Ashutosh Modi. Pre-trained language models as prior knowledge for playing text-based games. *arXiv preprint arXiv:2107.08408*, 2021.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33: 16857–16867, 2020.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep calm and explore: Language models for action generation in text-based games. *arXiv preprint arXiv:2010.02903*, 2020.

A Appendix

A.1 Prompts

Table 3: **Hopper**. An example of a prompt.

”x position: -0.00287 m, z position: 1.24741 m, y angle: -0.00328 rad, thigh angle: -0.00231 rad, leg angle: -0.00302 rad, foot angle: -0.00395 rad, x velocity: -0.00037 m/s, z velocity: 0.00124 m/s, y angular velocity: -0.00342 rad/s, thigh angular velocity: 0.00295 rad/s, leg angular velocity: 0.00157 rad/s, foot angular velocity: -0.00121 rad/s”

Table 4: **Walker2d**. An example of a prompt.

”x position: 0.00394 m, z position: 1.24658 m, y angle: -0.00189 rad, right thigh angle: 0.00481 rad, right leg angle: -0.00054 rad, right foot angle: -0.00291 rad, left thigh angle: 0.00076 rad, left leg angle: -0.0006 rad, left foot angle: 0.00068 rad, x velocity: 0.00244 m/s, z velocity: -0.00086 m/s, y angular velocity: -0.00418 rad/s, right thigh angular velocity: 0.00155 rad/s, right leg angular velocity: -0.00413 rad/s, right foot angular velocity: 0.00015 rad/s, left thigh angular velocity: -0.00351 rad/s, left leg angular velocity: 0.00191 rad/s, left foot angular velocity: -0.00086 rad/s”

Table 5: **HalfCheetah**. An example of a prompt.

”x position: 0.01921 m, z position: 0.05191 m, y angle: -0.06347 rad, back thigh angle: 0.07034 rad, back shin angle: -0.07455 rad, back foot angle: -0.04487 rad, front thigh angle: 0.01305 rad, front shin angle: -0.05206 rad, front foot angle: 0.04236 rad, x velocity: 0.0732 m/s, z velocity: 0.00765 m/s, y angular velocity: -0.03904 rad/s, back thigh angular velocity: -0.03949 rad/s, back shin angular velocity: -0.01819 rad/s, back foot angular velocity: 0.07899 rad/s, front thigh angular velocity: 0.2711 rad/s, front shin angular velocity: -0.0694 rad/s, front foot angular velocity: 0.02239 rad/s”

Table 6: **FinRL**. An example of a prompt. Since the resulting prompt is too big for an employed LM, we omitted the repetitive denotations for each ticker. The numbers from left to right: close price, amount, macd, rsi_30, cci_30, and dx_30 as in the original FinRL work (Liu et al., 2021c).

”Current balance: 1000000.0, AAPL: 2.79591 0.0 0.00373 100.0 66.66666 100.0, MSFT: 15.80062 0.0 0.0143 100.0 66.66666 100.0, JPM: 33.68094 0.0 0.04309 100.0 66.66666 100.0, V: 32.5144 0.0 0.03483 100.0 66.66666 100.0, RTX: 12.78609 0.0 0.01116 100.0 66.66666 100.0, PG: 48.04326 0.0 0.03592 100.0 66.66666 100.0, GS: 14.52728 0.0 0.00677 100.0 -66.66666 100.0, NKE: 20.5975 0.0 0.02376 100.0 66.66666 100.0, DIS: 72.84446 0.0 0.04464 100.0 66.66666 100.0, AXP: 17.90945 0.0 0.01848 100.0 66.66666 100.0, HD: 59.60556 0.0 0.04913 100.0 66.66666 100.0, INTC: 10.5273 0.0 0.00839 100.0 66.66666 100.0, WMT: 42.26554 0.0 0.01282 100.0 66.66666 100.0, IBM: 23.51501 0.0 0.00337 100.0 -66.66666 100.0, MRK: 14.13562 0.0 0.00435 100.0 66.66666 100.0, UNH: 44.2012 0.0 0.02427 100.0 66.66666 100.0, KO: 42.6148 0.0 0.02665 100.0 66.66666 100.0, CAT: 20.1181 0.0 0.00874 100.0 66.66666 100.0, TRV: 15.47116 0.0 0.0152 100.0 66.66666 100.0, JNJ: 8.9564 0.0 0.0078 100.0 66.66666 100.0, CVX: 10.93811 0.0 0.00752 100.0 66.66666 100.0, MCD: 43.36128 0.0 0.01518 100.0 66.66666 100.0, VZ: 25.85366 0.0 0.01425 100.0 66.66666 100.0, CSCO: 33.45659 0.0 0.0 100.0 66.66666 100.0, XOM: 23.21503 0.0 0.01869 100.0 66.66666 100.0, BA: 12.24426 0.0 0.00509 100.0 66.66666 100.0, MMM: 18.18748 0.0 0.00872 100.0 66.66666 100.0, PFE: 19.18782 0.0 0.01483 100.0 66.66666 100.0, WBA: 42.72148 0.0 0.01877 100.0 66.66666 100.0, DD: 53.48384 0.0 0.0266 100.0 66.66666 100.0”