Prompts and Pre-Trained Language Models for Offline Reinforcement Learning

Denis Tarasov*  
Tinkoff, Russia  
ext.dtarasov@tinkoff.ru

Vladislav Kurenkov*  
Tinkoff, Russia  
v.kurenkov@tinkoff.ai

Sergey Kolesnikov  
Tinkoff, Russia  
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 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.

2 Background
2.1 Offline RL
Reinforcement Learning (RL) is commonly defined by Markov Decision Process (MDP) with a tuple \((S, A, P, r, \gamma)\), 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 \(\gamma\) is a discount factor. The goal of the agent is to obtain a policy \(\pi(s, a)\) that maximizes \(E_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \right]\) 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 \(\pi_\beta\) 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).

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 and 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

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 and 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
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.

<table>
<thead>
<tr>
<th>Task</th>
<th>BC</th>
<th>CQL</th>
<th>PLAS</th>
<th>BCQ</th>
<th>TD3+BC</th>
<th>TD3+BC LM-Finetune</th>
<th>TD3+BC LM-No-Tune</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopper-v3-L-99</td>
<td>515</td>
<td>527</td>
<td>527</td>
<td>545</td>
<td>660</td>
<td>762</td>
<td>654</td>
</tr>
<tr>
<td>Walker2d-v3-L-99</td>
<td>1749</td>
<td>2370</td>
<td>42</td>
<td>1407</td>
<td>2480</td>
<td>2669</td>
<td>2564</td>
</tr>
<tr>
<td>HalfCheetah-v3-L-99</td>
<td>3260</td>
<td>3792</td>
<td>3451</td>
<td>3363</td>
<td>4171</td>
<td>4084</td>
<td>4171</td>
</tr>
</tbody>
</table>

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 and 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 $3 \times 10^5$ 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.

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.

**Sensitivity to Online Evaluation Budget** The preference between deep offline RL algorithms was noted to be dependent on the online evaluation budget available (Kurenkov and 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.

**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
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.

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

sufficient to achieve substantial improvements over the base model in most settings.

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


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


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: -0.00487 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 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 0.02370 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 0.0337 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, VZ: 25.85366 0.0 0.01425 100.0 66.66666 100.0, CVX: 33.45659 0.0 0.01483 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"
```