

756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809

A BROADER IMPACT

Designing a reward function is not a trivial challenge for many real-world problems where Reinforcement Learning can be useful. Moreover, in domains where interacting with the environment is limited or costly, sample inefficiency of RL algorithms can become a bottleneck for learning a policy. While domain experts can hand-engineer intrinsic rewards to aid the RL agent training, it is unreasonable to expect them to design useful reward shaping functions for each independent task. As a single step towards relaxing this expectation, we aim to leverage the universality of using Large Language Models (which can loosely be referred to “*Jack of all trades, master of none*”) to provide useful guidance for training the RL agent. Hence, one of the direct impacts of our work is tapping into the potential of LLMs for obtaining useful guidance across multiple tasks thereby reducing the cognitive load on domain experts. Moreover, we hope that our work opens up future possibilities of exploring more domains where LLMs can assist in aiding the downstream learning process.

B ENVIRONMENTS

B.1 BABYAI

In the BabyAI suite of environments, the agent has to reach the goal location in a fixed number of steps (environment max steps is set to 100). This platform relies on a gridworld environment (MiniGrid) to generate a set of complex environments, and includes challenging domains to test sample efficiency of training RL agents. Each gridworld environment is populated with the agent and objects such as doors, keys, and lava, which are placed in varying sized grids. Each grid is procedurally generated, i.e., the object placements can vary at the start of each episode. However, using a fixed environment seed, the layout stays the same for each episode run. The doors in the environment can also be locked, and can be opened by first picking up the key first. Objects such as lava can not be crossed by the agent, and lead to episode termination if the agent steps on one of these.

Observation Space: By default, the environment offers an observation space of shape (7,7,3) which is a partial view of the complete environment grid, and has been shown to accelerate the RL agent’s training (Chevalier-Boisvert et al., 2018). This observation space consists of a hierarchical mapping for 3 matrices each of size 7x7. The first contains the object-id, the second contains the property (color) of the object, and the third consists of the state of the object (e.g., if the door is open or locked).

Action Space: The agent’s action space consists of : turn left, turn right, move forward, pick up an object, drop an object, toggle/activate an object, and done. For each environment, some of these actions are unused, and we refer readers to the specific environment’s documentation for the respective details.

Extrinsic/Environment Rewards: If the agent reaches the goal location in N steps, the total reward is calculated as $R = 1 - 0.9*(N/H)$ where H is the maximum number of allowed steps (i.e., 100). Otherwise, the agent get a reward of 0 for that episode.

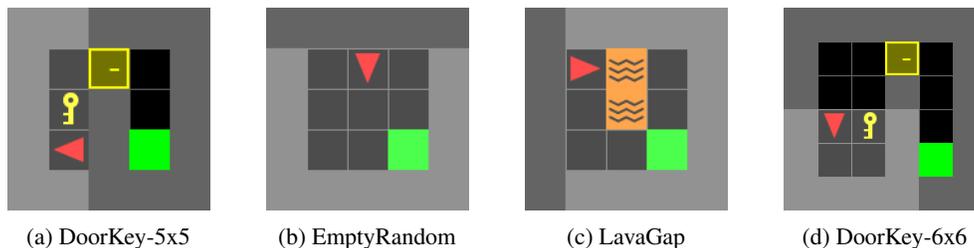


Figure 5: Environment Layouts from the BabyAI environment suite used for experiments.

B.1.1 DOORKEY

Description: The agent has to pick up the key to unlock the door and then reach to the goal location shown by the green square.

Mission Space: “use the key to open the door and then get to the goal”

Action Space: turn left, turn right, move forward, pick up an object, toggle/activate an object

B.1.2 EMPTY-RANDOM

Description: The agent has to reach the goal location shown by the green square in a completely empty room. The agent’s starting position can be varied by varying the environment seed.

Mission Space: “get to the green goal square”

Action Space: turn left, turn right, move forward

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

B.1.3 LAVAGAP

Description: The agent has to reach to the goal location shown by the green square by avoiding all lava tiles, where the episode will terminate immediately.

Mission Space: “avoid the lava and get to the green goal square”

Action Space: turn left, turn right, move forward

B.2 HOUSEHOLD

Description: The agent’s goal is to reach the final destination (green block). for that the robot (red triangle) has to pick up the red key, charge itself at the charging dock (purple block), and open the red door.

Observation Space: The household environment is an 15x15 grid. The state is represented by a multi-dimensional vector where each index corresponds to a specific grid location, and each value represents the type of object at that location. The state vector also includes boolean flags for whether the agent has picked up the right key, opened the door, and whether the agent is charged.

Action Space: up, down, left, right.

Reward: The reward function is a sparse binary reward, assigning a value of 0 to all states except the goal state.

B.3 MARIO

Description: The agent’s goal is to open the door upstairs. To do this, it must pick up both keys located downstairs. One key is hidden under a red rock. The ladder is worn out and will break after one use, so the agent can only go down through the tube, but not back up.

Observation Space: The Mario environment is an 8x11 grid. The state is represented by a multi-dimensional vector where each index corresponds to a specific grid location, and each value represents the type of object at that location. The state vector also includes boolean flags for whether the agent has visited the bottom, picked up the key, picked up the hidden key, and returned to the upper level.

Action Space: up, down, left, right.

Reward: The reward function is a sparse binary reward, assigning a value of 0 to all states except the goal state.

B.4 MINECRAFT

Description: The agent’s goal is to build a ladder using a plank and a stick. To do this, it needs to collect raw wood, take it to the processing unit to get processed wood, then make a plank at workshop1 and a stick at workshop2. Finally, both are used to build the ladder at workshop3.

Observation Space: This environment is an 10x15 grid. The state is represented by a multi-dimensional vector where each index corresponds to a specific grid location, and each value represents the type of object at that location. The state vector also includes boolean flags for whether a stick, plank, or ladder has been made, along with the number of processed wood pieces.

Action Space: up, down, left, right.

Reward: The reward function is a sparse binary reward, assigning a value of 0 to all states except the goal state.

C ALGORITHM AND EXPERIMENT DETAILS

All the experiments were run on an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, with GeForce GTX 1080 GPU.

C.1 ALGORITHM

We present the algorithm for our framework below:

Algorithm 1 Training an RL policy π_θ using π_{LLM}

Require: Input π_{LLM}

Ensure: Output π_θ

- 1: Initialize parameters θ , dataset \mathcal{D}
- 2: // EXPLORATION PHASE
- 3: **for** each iteration **do**
- 4: Store transitions $\mathcal{D} \leftarrow (s, a, s', r) \sim \pi_\theta$
- 5: Relabel dataset with shaped rewards $\mathcal{D}' \leftarrow \mathcal{D}, \pi_{LLM}$
- 6: **end for**
- 7: // REINFORCEMENT LEARNING PHASE
- 8: **for** each gradient step **do**
- 9: Sample transitions $\{\tau_j\}_{j=1}^{\mathcal{D}'} \sim \mathcal{D}'$
- 10: train π_θ using $\{\tau_j\}$
- 11: **end for**
- 12: **return** π_θ

C.2 HYPERPARAMETERS

C.2.1 RQ1 EXPERIMENTS

For RQ1 results, we set the temperature to 0.5 for directly prompting LLMs to generate the abstract plan. Since the LLM+verifier framework generates the plan step-by-step, we bound the maximum number of steps the LLM can take to reach the goal to 20 for *Household* and *MineCraft* and 30 for *BabyAI* and *Mario* environment. We also limit the maximum number of back-prompts allowed at any step to 5 for *Household* and *MineCraft* and 10 for *BabyAI* and *Mario* environment.

C.2.2 RQ2 EXPERIMENTS

For the Household, Mario, and Minecraft environments, we used the standard Q-learning algorithm with an ϵ -greedy policy. Initially, ϵ is set to 1 and annealed to 0.05 over the course of training. When the agent achieves a subgoal (as determined by the LLM plan), ϵ decays by a factor of 0.995, and it decays by a factor of 0.95 when the main goal is reached. The other hyperparameters are as follows: maximum training steps = 5e6, discount factor (γ) = 0.99, buffer size = 5e5, and batch size = 64.

The hyperparameters used for all PPO and A2C training experiments are listed in Table [3](#).

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

Table 3: Hyperparameters for all RL training experiments.

Hyperparameter	Value
# of training steps	1e6
# of epochs	4
batch size	256
discount	0.99
learning rate	0.001
gae lambda	0.95
entropy coefficient	0.01
value loss coefficient	0.5
max gradient norm	0.5
optimizer epsilon	1e-08
optimizer alpha	0.99
clip epsilon	0.2

1026 D PROMPTS

1027

1028 D.1 RQ1: DIRECT LLM PROMPTS

1029

1030 Here, we present the exact prompt for querying the entire policy at once for the DoorKey 5x5
1031 environment, the Mario environment, the Household environment, and the Mine-craft environment.

1032

1033 **Direct Prompt for DoorKey 5x5 environment**

1034

1035 **[PROMPT]** You are tasked with solving a 3x3 maze where you will encounter objects like
1036 a key and a door along with walls. Your task is 'use the key to open the door and then get to
1037 the goal'. You can be facing in any of the four directions. To move in any direction, to pick
1038 up the key, and to open the door, you need to face in the correct direction. The available
1039 actions at each step are 'turn left', 'turn right', 'move forward', 'pickup key', 'open door'.
1040 The current maze looks like this:

1041

1042 unseen door unseen
1043 key wall unseen
1044 agent wall goal

1045

1046 You (agent) are currently facing left.

1047 What is the sequence of actions you will take to reach the goal? Output as a comma
1048 separated list. Do not include anything else in your response.

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

Direct Prompt for Mario environment**[PROMPT]**

Here is a pddl domain, a planning problem. Provide the sequence of actions you will take to reach the goal. Provide only the pddl syntax for the plan where the action is represented as (ACTION-NAME OBJECTS). Do not provide anything else in your response.

domain pddl

```
(define (domain Mario)
  (:requirements :strips :typing)
  (:types key - object)
  (:predicates (has-key)
               (has-hidden-key)
               (at-upper-platform)
               (at-bottom)
               (at-upper-platform-with-key)
               (at-upper-platform-with-hidden-key)
               (door-open))
  (:action go_down_the_tube
           :parameters ()
           :precondition (and (at-upper-platform))
           :effect (and (at-bottom) ))
  (:action pickup_key
           :parameters ()
           :precondition (and (at-bottom))
           :effect (and (has-key) ))
  (:action pickup_hidden_key
           :parameters ()
           :precondition (and (at-bottom))
           :effect (and (has-hidden-key) ))
  (:action go_up_the_ladder
           :parameters ()
           :precondition (and (has-key) (has-hidden-key)
                              (at-bottom))
           :effect (and (at-upper-platform-with-key)
                              (at-upper-platform-with-hidden-key) ))
  (:action unlock_door
           :parameters ()
           :precondition (and (at-upper-platform-with-key)
                              (at-upper-platform-with-hidden-key))
           :effect (and (door-open)))
)
```

problem pddl

```
(define (problem prob)
  (:domain Mario)
  (:objects
   )
  (:init
   (at-upper-platform))
  (:goal
   (and (door-open)))
)
```

Direct Prompt for Household environment**[PROMPT]**

Here is a pddl domain, a planning problem. Provide the sequence of actions you will take to reach the goal. Provide only the pddl syntax for the plan where the action is represented as (ACTION-NAME OBJECTS). Do not provide anything else in your response.

domain pddl

```
(define (domain household)
  (:requirements :strips :typing :negative-preconditions)
  (:types key door - object)
  (:predicates (key-picked)
               (holding-key)
               (door-opened)
               (at-starting-location)
               (charged)
               (at-destination))
  (:action get_key
    :parameters ()
    :precondition (and (not (holding-key)))
    :effect (and (key-picked) (holding-key)))
  (:action open_door
    :parameters ()
    :precondition (and (not (door-opened))
                       (holding-key) (key-picked))
    :effect (and (door-opened) (not (holding-key))
                 (not (key-picked))))
  (:action is_charged
    :parameters ()
    :precondition (and (door-opened))
    :effect (and (charged)))
  (:action goal
    :parameters ()
    :precondition (and (charged) )
    :effect (and (at-destination)))
)
```

problem pddl

```
(define (problem prob)
  (:domain household)
  (:objects
  )
  (:init
    (at-starting-location))
  (:goal
    (and (at-destination)))
))
```

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

Direct Prompt for Mine-Craft environment**[PROMPT]**

Here is a pddl domain, a planning problem. Provide the sequence of actions you will take to reach the goal. Provide only the pddl syntax for the plan where the action is represented as (ACTION-NAME OBJECTS). Do not provide anything else in your response.

domain pddl

```
(define (domain minecraft)
  (:requirements :strips :typing :negative-preconditions)
  (:types wood - object)
  (:predicates
    (wood-picked ?w - wood)
    (wood-processed ?w - wood)
    (at-starting-location)
    (plank_made)
    (stick_made)
    (ladder_made)
    (processed-to-plank ?w - wood)
    (processed-to-stick ?w - wood))
  (:action get_wood
    :parameters (?w - wood)
    :precondition (not (wood-picked ?w))
    :effect (and (wood-picked ?w)))
  (:action get_processed_wood
    :parameters (?w - wood)
    :precondition (and (wood-picked ?w)
      (not (wood-processed ?w)))
    :effect (and (wood-processed ?w)))
  (:action make_plank
    :parameters (?w - wood)
    :precondition (and (wood-processed ?w)
      (not (processed-to-plank ?w))
      (not (processed-to-stick ?w)))
    :effect (and (processed-to-plank ?w) (plank_made)))
  (:action make_stick
    :parameters (?w - wood)
    :precondition (and (wood-processed ?w)
      (not (processed-to-stick ?w))
      (not (processed-to-plank ?w)))
    :effect (and (processed-to-stick ?w) (stick_made)))
  (:action make_ladder
    :parameters ()
    :precondition (and (stick_made) (plank_made))
    :effect (and (ladder_made)))
  )
)
```

problem pddl

```
(define (problem prob)
  (:domain minecraft)
  (:objects
    wood0 wood1 - wood)
  (:init
    (at-starting-location))
  (:goal
    (and (ladder_made)))
  ))
```

1242 D.2 RQ1: VERIFIER-AUGMENTED LLM PROMPTS
1243

1244 Here we show representative step-prompts and corresponding back prompts for DoorKey 5x5
1245 environment, Mario environment, Household environment and Mine-Craft environment.

1246 **Step-prompt and Back-Prompt for DoorKey 5x5 environment**
1247

1248
1249 **[STEP PROMPTING]**

1250 You are tasked with solving a 3x3 maze where you will encounter objects like a key and a
1251 door along with walls. Your task is ‘use the key to open the door and then get to the goal’.
1252 You can be facing in any of the four directions. To move in any direction, to pick up the
1253 key, and to open the door, you need to face in the correct direction. You will be given a
1254 description of the maze at every step and you need to choose the next action to take. The
1255 available actions are ‘turn left’, ‘turn right’, ‘move forward’, ‘pickup key’, ‘open door’.
1256 The current maze looks like this:

1257
1258 unseen door unseen
1259 key wall unseen
1260 agent wall goal

1261 You (agent) are currently facing left.

1262 What is the next action that the agent should take? Only choose from the list of available
1263 actions. Do not include anything else in your response. For example, if you choose ‘move
1264 forward’, then only write ‘move forward’ in your response.

1265 **LLM Response: pickup key**

1266 **[BACK PROMPTING]**

1267 Information: You cannot ‘pickup key’ in this state as you are not facing the key. Please
1268 choose another action. The following actions are feasible in this state: [‘turn right’, ‘turn
1269 left’].

1270 The current maze looks like this:

1271
1272 unseen door unseen
1273 key wall unseen
1274 agent wall goal

1275 You (agent) are currently facing left.

1276 What is the next action that the agent should take? Only choose from the list of available
1277 actions. Do not include anything else in your response. For example, if you choose ‘move
1278 forward’, then only write ‘move forward’ in your response. You have already tried the
1279 following actions: pickup key. Please choose another action.

1280 **LLM Response: turn right**

1281 ...
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

Step-prompt and Back-Prompt for Mario environment**[STEP PROMPTING]**

Here is a pddl domain, a planning problem. Provide only the next action for the query problem. Provide only the pddl syntax for the plan where the action is represented as (ACTION-NAME OBJECTS). Do not provide anything else in your response.

domain pddl

```
(define (domain Mario)
  (:requirements :strips :typing)
  (:types key - object)
  (:predicates (has-key)
               (has-hidden-key)
               (at-upper-platform)
               (at-bottom)
               (at-upper-platform-with-key)
               (at-upper-platform-with-hidden-key)
               (door-open))
  (:action go_down_the_tube
           :parameters ()
           :precondition (and (at-upper-platform))
           :effect (and (at-bottom) ))
  (:action pickup_key
           :parameters ()
           :precondition (and (at-bottom))
           :effect (and (has-key) ))
  (:action pickup_hidden_key
           :parameters ()
           :precondition (and (at-bottom))
           :effect (and (has-hidden-key) ))
  (:action go_up_the_ladder
           :parameters ()
           :precondition (and (has-key) (has-hidden-key)
                              (at-bottom))
           :effect (and (at-upper-platform-with-key)
                              (at-upper-platform-with-hidden-key) ))
  (:action unlock_door
           :parameters ()
           :precondition (and (at-upper-platform-with-key)
                              (at-upper-platform-with-hidden-key))
           :effect (and (door-open)))
)
```

problem pddl

```
(define (problem prob)
  (:domain Mario)
  (:objects
   )
  (:init
   (at-upper-platform))
  (:goal
   (and (door-open)))
)
```

LLM Response: *pickup_key*

[BACK PROMPTING]

Information: Your plan so far - []. Your response - *pickup_key*. The action provided is not feasible because you are still at upstairs. Choose a valid action from the list `['(pickup_hidden_key),(go_down_the_tube),(unlock_door),(pickup_key),(go_up_the_ladder)']`.

Step-prompt and Back-Prompt for Household environment**[STEP PROMPTING]**

Here is a pddl domain, a planning problem. Provide only the next action for the query problem. Provide only the pddl syntax for the plan where the action is represented as (ACTION-NAME OBJECTS). Do not provide anything else in your response.

domain pddl

```
(define (domain household)
  (:requirements :strips :typing :negative-preconditions)
  (:types key door - object)
  (:predicates (key-picked)
               (holding-key)
               (door-opened)
               (at-starting-location)
               (charged)
               (at-destination))
  (:action get_key
    :parameters ()
    :precondition (and (not (holding-key)))
    :effect (and (key-picked) (holding-key)))
  (:action open_door
    :parameters ()
    :precondition (and (not (door-opened))
                       (holding-key) (key-picked))
    :effect (and (door-opened) (not (holding-key))
                 (not (key-picked))))
  (:action is_charged
    :parameters ()
    :precondition (and (door-opened))
    :effect (and (charged)))
  (:action goal
    :parameters ()
    :precondition (and (charged) )
    :effect (and (at-destination)))
)
```

problem pddl

```
(define (problem prob)
  (:domain household)
  (:objects
  )
  (:init
    (at-starting-location))
  (:goal
    (and (at-destination)))
)
```

LLM Response: *open_door*

[BACK PROMPTING]

Information: Your plan so far - []. Your response - *open_door*. The action provided is not feasible because you do not have key with you. Choose a valid action from the list [*is_charged*], [*get_key*], [*open_door*], [*goal*].

...

Step-prompt and Back-Prompt for Mine-Craft environment**[STEP PROMPTING]**

Here is a pddl domain, a planning problem. Provide only the next action for the query problem. Provide only the pddl syntax for the plan where the action is represented as (ACTION-NAME OBJECTS). Do not provide anything else in your response.

domain pddl

```
(define (domain minecraft)
  (:requirements :strips :typing :negative-preconditions)
  (:types wood - object)
  (:predicates
    (wood-picked ?w - wood)
    (wood-processed ?w - wood)
    (at-starting-location)
    (plank_made)
    (stick_made)
    (ladder_made)
    (processed-to-plank ?w - wood)
    (processed-to-stick ?w - wood))
  (:action get_wood
    :parameters (?w - wood)
    :precondition (not (wood-picked ?w))
    :effect (and (wood-picked ?w)))
  (:action get_processed_wood
    :parameters (?w - wood)
    :precondition (and (wood-picked ?w)
      (not (wood-processed ?w)))
    :effect (and (wood-processed ?w)))
  (:action make_plank
    :parameters (?w - wood)
    :precondition (and (wood-processed ?w)
      (not (processed-to-plank ?w))
      (not (processed-to-stick ?w)))
    :effect (and (processed-to-plank ?w) (plank_made)))
  (:action make_stick
    :parameters (?w - wood)
    :precondition (and (wood-processed ?w)
      (not (processed-to-stick ?w))
      (not (processed-to-plank ?w)))
    :effect (and (processed-to-stick ?w) (stick_made)))
  (:action make_ladder
    :parameters ()
    :precondition (and (stick_made) (plank_made))
    :effect (and (ladder_made)))
)
```

problem pddl

```
(define (problem prob)
  (:domain minecraft)
  (:objects
    wood0 wood1 - wood)
  (:init
    (at-starting-location))
  (:goal
    (and (ladder_made)))
))
```

1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

Step-prompt and Back-Prompt for Mine-Craft environment (..continued)

LLM Response: *make_ladder*

[BACK PROMPTING]

Information: Your plan so far - [*get_wood, get_processed_wood, make_stick*].
Your response - *make_ladder*. The action provided is not feasible because you do not have plank. Choose a valid action from the list [*(get_wood), (make_ladder), (make_stick), (make_plank), (get_processed_wood)*].
...