Textworld AI Project

Text-based game agent through RL environment

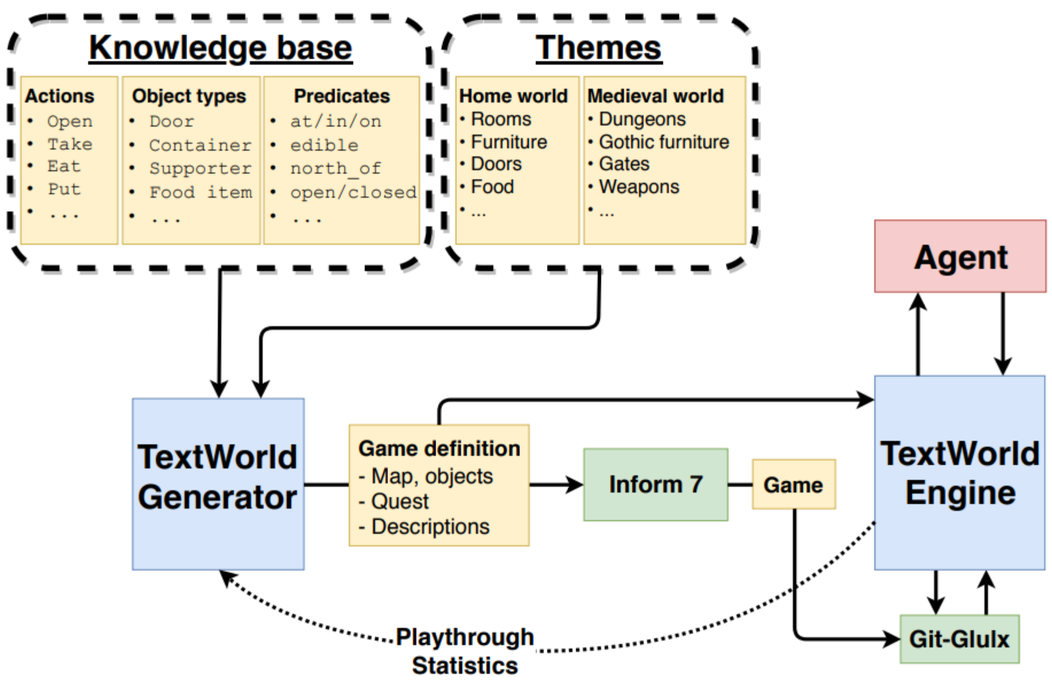
徐御倫 r06922138 林依蓁 R06944003 陳郁文 B05902071

Introduction

Text-based games are complex, interactive simulations in which text describes the game state and players make progress by entering text commands.

In this project, we created an agent using Microsoft Textworld framework to make a sequential decision-making model that takes text information as input and outputs text commands to progress through a game

Textworld Framework



Mythodology

Q-Learning Algorithm

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode):

  Initialize s

  Repeat (for each step of episode):

    Choose a from s using policy derived from Q

    Take action a, observe r, s'

    Update

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

  $s \leftarrow s'$;

Until s is terminal

In Q-learning the agent's experience consists of a sequence of distinct episodes. The available experience for an agent in an MDP environment can be defined by (s, a, r, r, γ)

- s: Environment State
- a: Set of Action
- t: State Transtion Function
- r: Reward
- γ: Epsilon discount factor

Implementation : Q-learning

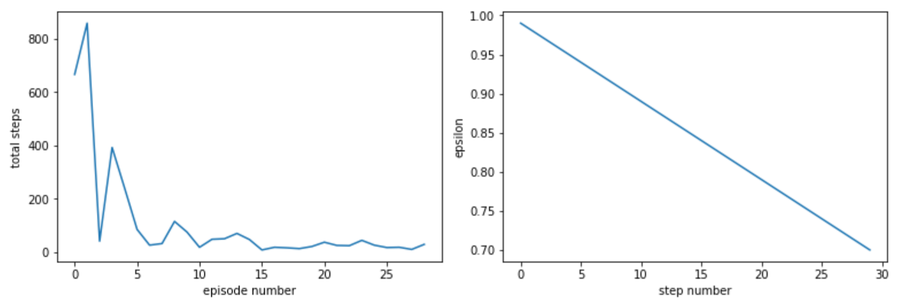
Experiment

- Take the room as the state. (There are 5 values)
- Get the candidate commands list with game_state.admissible_commands
- Get the immediate reward with env.compute_intermediate_reward() function, put (state, command, intermediate_reward) data into the Q-table, and keep updating the Q-table.
- 50% probability to randomly take a command from the command list, 50% probability to take the highest score action from the Q-table, and pass it to the env.step (command) function.
- Run 30 episodes / 1000 steps in a episode.
- **Compare the results for different epsilon formulation.**

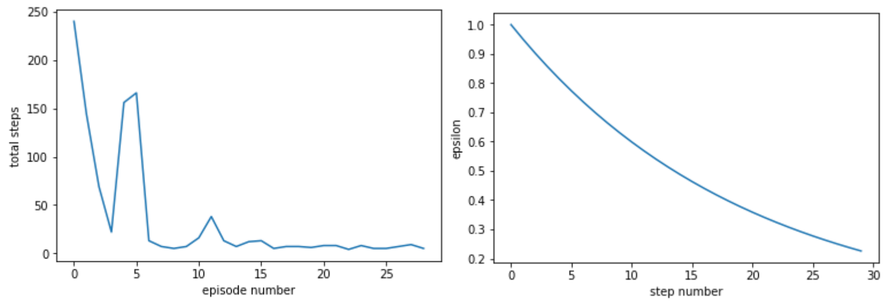
Result

- Exp1: epsilon = episode - 0.01; avg. steps: 137.6; avg. score: 1.0 / 1.
- Exp2: epsilon = math.pow(0.95, episode_num); avg. steps: 68.0; avg. score: 1.0 / 1.
- Exp3: epsilon = 1/math.pow(episode_num+1, 2); avg. steps: 18.1; avg. score: 1.0 / 1.

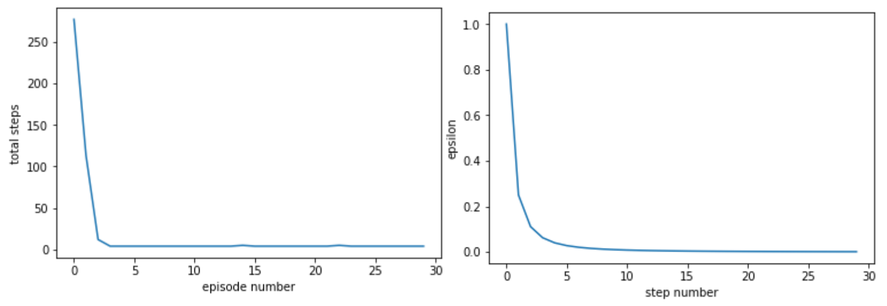
•Experiment 1
•avg. steps: 137.6;
•avg. score: 1.0 / 1.



•Experiment 2
•avg. steps: 68.0;
•avg. score: 1.0 / 1.



•Experiment 3
•avg. steps: 18.1;
•avg. score: 1.0 / 1.



Conclusion

•Using the Q-learning algorithm to train the agent performed very well. After about 10 episode training, the number of steps to complete the game can converge to dozens of steps, close to the ability of a human newbie.

•The conclusion is that when the formula of epsilon is 1 / t ^ 2, the performance is the best and the training process is the most stable.