ATAI

Interactive Fiction Games: A Colossal Adventure

Homework 1

Arnaud Le Doeuff, Ignacio Dorado, Jorge Orbegozo
28/09/2020

Table des matières

I. Introduction

Interactive fictions (IF), or text-adventures, are games in which a player interacts with a world entirely through textual descriptions and text commands. These games are typically structured as puzzles or quests where a player observes textual descriptions of a simulated world, executes an action inputting a natural language command and gets a change in the world as a reward to progress in the game.

The goal is to test the efficiency of autonomous Reinforcement Learning agents playing IF games. This problem does not only involve sequential decision making, but also natural language processing. Before we go deeper into the chosen approach, we present the main challenges to face.

Reinforcement Learning algorithms suit really well sequential decision making problems but are not usually used for text-based tasks, which makes the action space grow at a combinatorial range. A 7 word sentence using a vocabulary size of 700 can be formed in 700^4 (240 billion) different sentences, which is not a feasible exploring space. Also, the game may only recognise half of those sentences, and, out of them, only a few will make a change in the environment.

IF games are meant to be played using commonsense reasoning, humans know how to interact with their environment and have the knowledge to pair the right verb with the right object in the right situation.

IF games usually have many different locations and require the player to travel between them without a map or any graphic support. We need to track every location along with every object and the connectivity between places, which sometimes is not Euclidean.

To tackle those challenges, we introduce Jericho, a learning environment specific for IF games.

II. Jericho

Jericho is an open source IF environment based on Python, which provides an OpenAI-Gym-like interface for connecting learning agents with IF games. It supports a variety of game genres of human-made IF, for example, Sci-Fi, horror and mystery. Games are selected from Infocom. Following, we present the main features Jericho provides to make IF games more accessible to existing agents:

Jericho can load and save game states that allow planning algorithms like Monte-Carlo Tree Search. Jericho provides the option to seed the game's random number generation for replicability. Supported games use a point-based scoring system, which serves as the agent's reward.

- Template-Based Action Generation: Jericho can extract the game-specific vocabulary and action templates. First, the agent takes an action template containing up to blanks (i.e. *throw* __ *to* __), and then it fills the gaps with words of the game vocabulary. This really helps reduce the combinatorial action space. With Jericho's vocabulary, you're guaranteed to not miss crucial words.
- World Object Tree: Is a representation of the game state, is used to codify the relationship between objects and locations in the game world. An object has a parent, children and siblings. For example, the player object has his location as parent and his inventory items as children.
- Fixed random seed to enforce determinism: this opens up an opportunity to use algorithms like Go-Explore, which requires a deterministic state transition function.
- Load/save functionality: this feature allows us to use planning algorithms like Monte-Carlo Tree Search, since we can restore previous states of the game.
- Identifying Valid Actions: These are actions that generate changes in the game state. Jericho can detect valid actions by executing a candidate action and providing feedback on the success or failure of an agent's last action to effect a change in the game state (world-object-tree). Jericho is able to perform a search a identify all valid actions, this search is implemented as follows:

Algorithm 1 Procedure for Identifying Valid Actions

$\mathcal{E} \leftarrow$ Jericho environment
$\mathcal{T} \leftarrow$ Set of action templates
$o \leftarrow$ Textual observation
$\mathcal{P} \leftarrow \{p_1 \dots p_n\}$ Interactive objects identified with noun-phrase extraction or world object tree.
$Y \leftarrow \varnothing$ List of valid actions
$s \leftarrow \mathcal{E}.save()$ – Save current game state
for template $u \in \mathcal{T}$ **do**
    for all combinations $p_1, p_2 \in \mathcal{P}$ **do**
        Action $a \leftarrow u \Leftarrow p_1, p_2$
        if $\mathcal{E}.world\_changed(\mathcal{E}.step(a))$ **then**
            $Y \leftarrow Y \cup a$
            $\mathcal{E}.load(s)$ – Restore saved game state
return Y

III. Algorithms

In this section, three agents are presented:

- Choice-based single-game agent (DRRN)
- Parser-based single-game agent (TDQN)
- Parser-based general-game agent (NAIL)

Single-game agents are trained and evaluated on the same game, and general-game agents are trained and evaluated on unseen games.

Common Input Representation: The inputs are vectors that are converted with an encoder using the following process: Observations are tokenized by a SentencePiece model using a vocabulary trained on strings extracted from sessions of humans playing IF games. This are processed by separate GRU encoders for the narrative. The outputs of these encoders are concatenated into a vector. DRRN and TDQN build on this representation.

DRRN: Deep Reinforcement Relevance Network. Algorithm designed for games based in choices and present a set of valid actions at every game state. GRU is used to codify all valid actions in a vector and then this vector and the observation vector are concatenated. Using combined vector, DRRN calculates a Q-Value for every valid action. The network is updated by sampling a minibatch of transitions. DRRN uses Jericho's handicaps 2 and 5.

TDQN: Agent for parser-based games, that takes from a pre-defined set of verbs and objects and handles a combinatorial action space by generating verb-objects actions. Template-DQN is an extension of LSTM-DQN, which takes

templates instead of only verbs. TDQN estimate Q-values for the tree outputs, template and two words that fill the template's blanks. To help this agent a supervised binary-cross entropy loss is introduced. The idea behind this loss is to nudge the agent towards valid templates and words. TDQN uses the same handicaps than DRRN.

NAIL: General-game agent designed to score the maximum as possible in a single episode interaction of the game. NAIL uses no handicaps, a set of manually-created heuristics is used to build a map of objects and locations, and with that NAIL know if an action is valid or invalid. NAIL uses a web-based language to decide how to interact with the objects.

IV. Experiments

a. Results

The agents (TDQN & DRRN) were evaluated across a diverse set of 32 games, the aim of creating a reproducible benchmark to help the community track progress and move the state of the art. The two learning agents were compared with to two non-learning baseline agents:

- RAND: a random agent that picks randomly from a set of 12 common IF actions at each step
- NAIL: a competition-winning heuristic-based IF agent

A completion rate of 100 percent means finishing the game with maximum score. After the evaluation of the set of games, a completion rate is attributed for each agent:

Agent	RAND	NAIL	TDQN	DRRN
Completion rate (%)	1.8	4.9	6.1	10.7

TDQN and DRRN accumulate significantly higher scores than the other agents.

The success of these learning agents demonstrates Jericho is effective at reducing the difficulty of IF games and making them more accessible for RL agents to learn and improve language-based skills.

Jericho supports a variety of games, covering a diverse set of structures and genres. These games were categorized into three difficulty tiers: **Possible games**, **Difficult games** and **extreme games**. These categories of difficulties were determined with different criteria: Template action space, solution length, average steps per reward, stochastic, dialog, darkness, nonstandard actions, inventory limit.

b. Example with Zork1

Final reported scores are an average over 5 runs of each

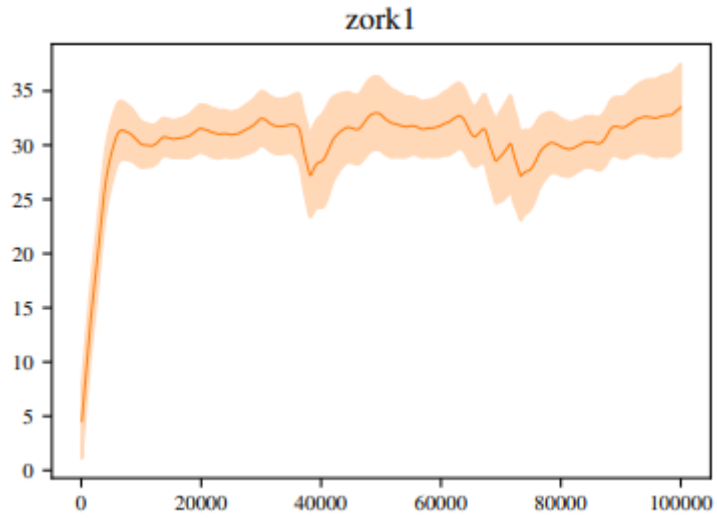
algorithm.

Let's see the raw score of each agent for the games “Zork1”.

Game	|T |	|V|	RAND	NAIL	TDQN	DRRN	Max Score
Zork 1	237	697	0	10.3	9.9	32.6	350

|T | denotes the number of templates and |V| is the size of the parser’s vocabulary. The Max score is the score to finnish the game.

Episode score as a function of training steps for DRRN. Shaded regions denote standard deviation across five independent runs for the game:



V. Conclusion

IF games are a challenge even for human players, Jericho has proven to be a great experimental platform to reduce the problem complexity. Thanks to Jericho templates and vocabulary extraction, Template-DQN agent was introduced. The fact that DRRN, the choice-based agent, did a better job than TQDN, reveals the difficulty of language generation.

NAIL algorithm showed worse performance than reinforcement learning agents. However, DRRN and TDQN were trained and evaluated on individual games. This leads us to conclude that obtaining a real general-purposes IF agent is still a greater challenge.

Also, there is still several work to do regarding template-based agent: TDQN algorithm computes independent Q-values for words and templates, conditional generation is an improvement yet to be explored.

VI. References

- "Interactive Fiction Games: A Colossal Adventure": https://arxiv.org/pdf/1909.05398.pdf
- Jericho github: https://github.com/microsoft/jericho
- "Deep Reinforcement Learning with a Natural Language Action Space (DRRN) ": https://arxiv.org/abs/1511.04636
- "Language Understanding for Text-based Games Using Deep Reinforcement Learning (LSTM-DQN)": https://arxiv.org/abs/1506.08941
- "NAIL: A General Interactive Fiction Agent": https://arxiv.org/abs/1902.04259
- DRRN and TDQN implementation: https://github.com/microsoft/tdqn
- NAIL implemetation: https://github.com/microsoft/nail_agent

VII. Appendices

Raw scores across Jericho supported games.

Game	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	RAND	NAIL	TDQN	DRRN	MaxScore
905	82	296	0	0	0	0	1
acorncourt	151	343	0	0	1.6	**10**	30
advent†	189	786	36	36	36	36	350
adventureland	156	398	0	0	0	**20.6**	100
afflicted	146	762	0	0	1.3	**2.6**	75
anchor	260	2257	0	0	0	0	100
awaken	159	505	0	0	0	0	50
balances	156	452	0	**10**	4.8	**10**	51
deephome	173	760	1	**13.3**	1	1	300
detective	197	344	113.7	136.9	169	**197.8**	360
dragon	177	1049	0	**0.6**	-5.3	-3.5	25
enchanter	290	722	0	0	8.6	**20.0**	400
gold	200	728	0	3	**4.1**	0	100
inhumane	141	409	0	0.6	**0.7**	0	90
jewel	161	657	0	**1.6**	0	1.6	90
karn	178	615	0	1.2	0.7	**2.1**	170
library	173	510	0	0.9	6.3	**17**	30
ludicorp	187	503	13.2	8.4	6	**13.8**	150
moonlit	166	669	0	0	0	0	1
omniquest	207	460	0	5.6	**16.8**	5	50
pentari	155	472	0	0	17.4	**27.2**	70
reverb	183	526	0	0	0.3	**8.2**	50
snacktime	201	468	0	0	**9.7**	0	50
sorcerer	288	1013	5	5	5	**20.8**	400
spellbrkr	333	844	25	**40**	18.7	37.8	600
spirit	169	1112	**2.4**	1	0.6	0.8	250
temple	175	622	0	7.3	**7.9**	7.4	35
tryst205	197	871	0	2	0	**9.6**	350
yomomma	141	619	0	0	0	**0.4**	35
zenon	149	401	0	0	0	0	20
zork1	237	697	0	10.3	9.9	**32.6**	350
zork3	214	564	0.2	**1.8**	0	0.5	7
ztuu	186	607	0	0	4.9	**21.6**	100

Table 1: **Raw scores** across Jericho supported games. $|\mathcal{T}|$ denotes the number of templates and $|\mathcal{V}|$ is the size of the parser's vocabulary. †Advent starts with a score of 36.