



# FIRST TEXTWORLD PROBLEMS

Renat Aksitov

CS230 - Winter 2019 - Stanford University

## Introduction

Learning to play text games is an important task in the language understanding domain

Text-based games are structured as a "dialog" between a game engine and the player

TextWorld by Microsoft Research - a learning environment for training and evaluation of Reinforcement Learning agents on text-based games

*First TextWorld Problems: A Reinforcement and Language Learning Challenge* - a ML competition in the TextWorld framework

### First TextWorld Problems competition

- An agent acts within the house
- He tries to gather the ingredients in order to cook a delicious meal
- The agent must:
  - figure out the ingredients from a recipe
  - explore the house to collect them
  - once done, go to the kitchen to do the cooking

```

Your objective is to sit the tiny grape on the dusty bench in the luxurious steam room.

.- Unreasonably Hot Dish-Pit -.
This might come as a shock to you, but you've just moved into an unreasonably hot dish-pit. You begin looking for stuff.

A locked safe is here. You can make out a soaped down saucepan. You see a yellow passkey on the saucepan. I mean, just wow! Isn't TextWorld just the best?

There is an exit to the south. Don't worry, it is unblocked.

There is a chilled sandwich on the floor.

> take sandwich
Taken.

> inventory
You are carrying:
  a chilled sandwich
  a large stick of butter

> eat it
You eat the chilled sandwich. Not bad.

>

```

Fig. 1: An example game from TextWorld with a house-based theme.

## Data

**4440 different games:** each game is a data file for the TextWorld framework with the full description of the world)

**The games** in the training set have a different complexity, which is determined by the "skills" that the agent needs to learn in order to perform well in that game

**The skills** include the number of ingredients in the recipe, actions ("open", "cook", "cut" and "drop") and the number of locations in the game

**Handicap:** an agent could request additional information for playing the games at the cost of a score penalty

## Challenges

For AI agent to play IF games efficiently it needs to:

- master language understanding
- deal with a combinatorial action space
- perform efficient exploration
- have memory and capability for sequential decision-making

Most of the commercially available text-based games are **beyond capabilities** of the existing algorithmic approaches

## Approach

**Game representation:** sequential decision making problem, where at each time step the agent receives strings of text with the description of the current state and several strings with all possible actions

**The agent** chooses one of these actions with the goal of maximizing future rewards

### Q-Learning

- A *discounted* reward:  $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$
- Q-function - the expected reward when following a policy  $\pi(a|s)$
- Q-learning - an algorithm to find optimal Q-function:
 
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta_t \cdot (r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

## Model Architecture

**Based** on DRRN

**Uses** trainable, randomly initialized, embeddings

**Adds** prioritized buffer for more valuable or recent experiences

**Uses** RNN as a state encoder

**GRU** as a recurrent unit

### DRRN architecture

- Uses 2 separate embeddings for states and actions
- Applies feed-forward networks to the BOW of embeddings to obtain Q-values
- The final Q-value is a dot product of states and actions Q-values

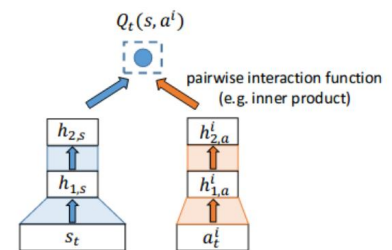


Fig. 2: DRRN architecture with 2 hidden layers for both states and actions

## Results

Model architecture	Easy (3)	Moderate (6)	Hard (10)	Episodes	Time, sec
Random baseline	1.2	0.8	0.3	n/a	n/a
DRRN-BASIC	3.0	2.5	1.5	120	362
DRRN-PRIOR	3.0	5.2	2.5	52	156
DRRN-GRU-BOTH	3.0	4.5	4.2	15	297
DRRN-SHARED-EMB	3.0	6.0	3.1	27	80
DRRN-GRU-STATE	3.0	6.0	8.0	8	245

## Future Work

**Generalize** to unseen games

**Use** target network in DQN

**Reduce** handicap

**Learn** epsilon in  $\epsilon$ -greedy

## Hyper parameters

Hyper Parameter	Value	Search Space
Optimizer	Adam	{ Adam; RMSProp }
Learning rate	0.001	{ 0.1 .. 0.00001 }
Dropout	0.9	{ 0.5 .. 1.0 }
Gamma	0.5	{ 0.4 .. 1.0 }
Replay buffer	10,000	{ 1,000 .. 100,000 }
RNN encoder	GRU	{ GRU; LSTM }
RNN hidden size	128	{ 32 .. 256 }
Grad norm	5.0	{ None; 1.0 .. 10.0 }