MiuLab / TC-Bot

User Simulation for Task-Completion Dialogues

#user-simulator #nlg #dialogue-agents #nlu #end-to-end

43 commits | 2 branches | 0 packages | 0 releases | 4 contributors | MIT

Branch: master | New pull request | Find file | Clone or download

yvchen update the reference — Latest commit 97f6a25 on 27 Sep 2017

imgs	Add all user simulator code	3 years ago
src	fix one parameter	3 years ago
.gitignore	fixed model name typo & add .gitignore	3 years ago
LICENSE	Create LICENSE	3 years ago
README.md	update the reference	3 years ago

README.md

End-to-End Task-Completion Neural Dialogue Systems

An implementation of the
End-to-End Task-Completion Neural Dialogue Systems and A User Simulator for Task-Completion Dialogues.

This document describes how to run the simulation and different dialogue agents (rule-based, command line, reinforcement learning). More instructions to plug in your customized agents or user simulators are in the Recipe section of the paper.

Content

Data

all the data is under this folder: ./src/deep_dialog/data

- Movie Knowledge Bases
  `movie_kb.1k.p` --- 94% success rate (for `user_goals_first_turn_template_subsets.v1.p`)
  `movie_kb.v2.p` --- 36% success rate (for `user_goals_first_turn_template_subsets.v1.p`)
- User Goals
  `user_goals_first_turn_template.v2.p` --- user goals extracted from the first user turn
  `user_goals_first_turn_template.part.movie.v1.p` --- a subset of user goals [Please use this one, the upper bound success rate on movie_kb.1k.json is 0.9765.]

- NLG Rule Template
  `dia_act_nl_pairs.v6.json` --- some predefined NLG rule templates for both User simulator and Agent.
- Dialog Act Intent
  `dia_acts.txt`
- Dialog Act Slot
  `slot_set.txt`

Parameter

Basic setting

`--agt` : the agent id
`--usr` : the user (simulator) id
`--max_turn` : maximum turns
`--episodes` : how many dialogues to run
`--slot_err_prob` : slot level err probability
`--slot_err_mode` : which kind of slot err mode
`--intent_err_prob` : intent level err probability

Data setting

`--movie_kb_path` : the movie kb path for agent side
`--goal_file_path` : the user goal file path for user simulator side

Model setting

`--dqn_hidden_size` : hidden size for RL (DQN) agent
`--batch_size` : batch size for DQN training
`--simulation_epoch_size` : how many dialogue to be simulated in one epoch
`--warm_start` : use rule policy to fill the experience replay buffer at the beginning
`--warm_start_epochs` : how many dialogues to run in the warm start

Display setting

`--run_mode` : 0 for display mode (NL); 1 for debug mode (Dia_Act); 2 for debug mode (Dia_Act and NL); >3 for no display (i.e. training)
`--act_level` : 0 for user simulator is Dia_Act level; 1 for user simulator is NL level
`--auto_suggest` : 0 for no auto_suggest; 1 for auto_suggest
`--cmd_input_mode` : 0 for NL input; 1 for Dia_Act input. (this parameter is for AgentCmd only)

Others

`--write_model_dir` : the directory to write the models
`--trained_model_path` : the path of the trained RL agent model; load the trained model for prediction purpose.

`--learning_phase` : train/test/all, default is all. You can split the user goal set into train and test set, or do not split (all); We introduce some randomness at the first sampled user action, even for the same user goal, the generated dialogue might be different.

Running Dialogue Agents

Rule Agent

```
python run.py --agt 5 --usr 1 --max_turn 40
              --episodes 150
              --movie_kb_path ./deep_dialog/data/movie_kb.1k.p
              --goal_file_path ./deep_dialog/data/user_goals_first_turn_template.part.movie.v1.p
              --intent_err_prob 0.00
              --slot_err_prob 0.00
              --episodes 500
              --act_level 0
```

Cmd Agent

NL Input

```
python run.py --agt 0 --usr 1 --max_turn 40
              --episodes 150
              --movie_kb_path ./deep_dialog/data/movie_kb.1k.p
              --goal_file_path ./deep_dialog/data/user_goals_first_turn_template.part.movie.v1.p
              --intent_err_prob 0.00
              --slot_err_prob 0.00
              --episodes 500
              --act_level 0
              --run_mode 0
              --cmd_input_mode 0
```

Dia_Act Input

```
python run.py --agt 0 --usr 1 --max_turn 40
              --episodes 150
              --movie_kb_path ./deep_dialog/data/movie_kb.1k.p
              --goal_file_path ./deep_dialog/data/user_goals_first_turn_template.part.movie.v1.p
              --intent_err_prob 0.00
              --slot_err_prob 0.00
              --episodes 500
              --act_level 0
              --run_mode 0
              --cmd_input_mode 1
```

End2End RL Agent

Train End2End RL Agent without NLU and NLG (with simulated noise in NLU)

```
python run.py --agt 9 --usr 1 --max_turn 40
              --movie_kb_path ./deep_dialog/data/movie_kb.1k.p
              --dqn_hidden_size 80
              --experience_replay_pool_size 1000
              --episodes 500
              --simulation_epoch_size 100
              --write_model_dir ./deep_dialog/checkpoints/rl_agent/
              --run_mode 3
              --act_level 0
              --slot_err_prob 0.00
              --intent_err_prob 0.00
              --batch_size 16
              --goal_file_path ./deep_dialog/data/user_goals_first_turn_template.part.movie.v1.p
              --warm_start 1
              --warm_start_epochs 120
```

Train End2End RL Agent with NLU and NLG

```
python run.py --agt 9 --usr 1 --max_turn 40
              --movie_kb_path ./deep_dialog/data/movie_kb.1k.p
              --dqn_hidden_size 80
              --experience_replay_pool_size 1000
              --episodes 500
              --simulation_epoch_size 100
              --write_model_dir ./deep_dialog/checkpoints/rl_agent/
              --run_mode 3
              --act_level 1
              --slot_err_prob 0.00
              --intent_err_prob 0.00
              --batch_size 16
              --goal_file_path ./deep_dialog/data/user_goals_first_turn_template.part.movie.v1.p
              --warm_start 1
              --warm_start_epochs 120
```

Test RL Agent with N dialogues:

```
python run.py --agt 9 --usr 1 --max_turn 40
              --movie_kb_path ./deep_dialog/data/movie_kb.1k.p
              --dqn_hidden_size 80
              --experience_replay_pool_size 1000
              --episodes 300
              --simulation_epoch_size 100
              --write_model_dir ./deep_dialog/checkpoints/rl_agent/
              --slot_err_prob 0.00
              --intent_err_prob 0.00
              --batch_size 16
              --goal_file_path ./deep_dialog/data/user_goals_first_turn_template.part.movie.v1.p
              --trained_model_path ./deep_dialog/checkpoints/rl_agent/noe2e/agt_9_478_500_0.98000.p
              --run_mode 3
```

Evaluation

To evaluate the performance of agents, three metrics are available: success rate, average reward, average turns. Here we show the learning curve with success rate.

1. Plotting Learning Curve `python draw_learning_curve.py --result_file ./deep_dialog/checkpoints/rl_agent/noe2e/agt_9_performance_records.json`
2. Pull out the numbers and draw the curves in Excel

Reference

Main papers to be cited

```
@inproceedings{li2017end,
  title={End-to-End Task-Completion Neural Dialogue Systems},
  author={Li, Xuijun and Chen, Yun-Nung and Li, Lihong and Gao, Jianfeng and Celikyilmaz, Asli},
  booktitle={Proceedings of The 8th International Joint Conference on Natural Language Processing},
  year={2017}
}

@article{li2016user,
  title={A User Simulator for Task-Completion Dialogues},
  author={Li, Xiujun and Lipton, Zachary C and Dhingra, Bhuwan and Li, Lihong and Gao, Jianfeng and Chen,
Yun-Nung},
  journal={arXiv preprint arXiv:1612.05688},
  year={2016}
}
```