ParlAI: A Dialog Research Software Platform (Facebook AI: 2018)

**[1. Introduction]**

**[2. Goals]**

* 대화(dialog) 모델 개발을 위한 단일화된 framework.
* 다양한 skills를 포함하는 일반적인(general) 대화.
* 사람들과 실제 대화.
* 공통의 일반적인 대화 모델을 통해. (Towards a common general dialog model)

**[3. General Properties of ParlAI]**

**[4. Worlds, Agents and Teachers]**

ParlAI의 주요 개념(클래스)는 worlds, agents, teachers이다.

|  |  |
| --- | --- |
| world | 환경(environment), 아주 심플한 것(두 에이전트가 대화)부터 복잡한 것(여러 개의에이전트가 상호적인 환경에서) 까지 다양하다. |
| agent | world에서 act(특별히 speak)할 수 있는 agent. agent는 또한 learner가 될 수 있다. (hard-coded bot, learner들이나 사람들과 상호작용할 수 있는, ex) Turker) |
| teacher | 가르치기 위해서 learner에게 말을 하는 agent의 유형 |

world와 agent를 정의하고 나서, main loop은 training, testing, displaying하기 위해 실행될 수 있다. world.parley()가 world의 one time step을 실행하는 함수이다.

<코드 예제, Fig 3>

|  |
| --- |
| Teacher = SquadTeacher(opt)  Agent = MyAgent(opt)  world = World(opt, [teacher, agent])  for I in range(num\_exs):  world.parley()  print(world.display()) |

|  |
| --- |
| Def parley(self):  For agent in self.agents:  Act = agent.act()  For other\_agent in self.agents:  If other\_agent != agent:  Other\_agent.observe(act) |

<output, Fig 4>스크린샷이(가) 표시된 사진

자동 생성된 설명

**[5. Actions and Observations]**

모든 agent들(teachers를 포함해서)은 한가지 포맷을 통해 서로에게 말을 건다. => observation/action object (a python dict) – Fig 5.

Text, labels, reward를 통해 agents간에 pass된다.

Talking(Acting), Listenging(observing)

Agent.act()에서 리턴된 값이 agent.observe()로 pass된다. – Fig3

메세지의 fields

|  |
| --- |
| * Text : * Id * Reward * Episode\_done |

Supervised datasets에는 additional fields들이 있다.

|  |
| --- |
| * Label * Label\_candidate * Text\_candidate * Metrics * Image |

각 fields들은 Dataset에 따라 optional하다.

Parl AI training set에서 전형적인 exchange – Fig.6

스크린샷이(가) 표시된 사진

자동 생성된 설명

**[6. Code Structure]**

ParlAI codebase는 5개의 주요 디렉토리가 있다.

|  |  |
| --- | --- |
| * core | 플랫폼의 주요 코드 |
| * agents | Worlds와 task로 상호작용할 수 있는 agents들을 포함 (learning models) |
| * examples | 다른 main(display data, training과 evaluation)의 example들을 포함 |
| * tasks | ParlAI에서 가능한 다른 tasks들에 대한 코드를 포함 |
| * mturk | Mechanical Turk를 세팅하기 위한 코드와 샘플 MTurk tasks 포함 |

**[6.1 Core]**

Core 라이브러리는 다음과 같은 파일을 포함하고 있다.

|  |  |
| --- | --- |
| * agents.py | 모든 agents에 관해 Agent 베이스 클래스를 정의한다. Observer(), act() 메소드 정의 |
| * dialog\_teacher.py | 고쳐진 chat log로 다이얼로그를 하는 teacher class의 base |
| * worlds.py | Base world 클래스를 정의한다. |
| * dict.py | Language dictionary를 만들기 위한 코드 |
| * metrics.py | exact match를 계산한다. F1과 평가를 위한 ranking metrics |
| * params.py | ParlAI의 command line arguments를 interprete하기 위한 argparse를 사용한다. |

**[6.2 Agents]**

Agents 디렉토리는 머신러닝 agents를 포함하고 있다. 아래 디렉토리에서 available하다.

|  |  |
| --- | --- |
| * **drqa** | Attentive LSTM 모델인 DrQA는 파이토치로 구현되어 있다. SQuAD에서 좋은 결과가 나온다. |
| * **memnn** | End-toend 메모리 network를 위한 코드. Lua Torch로 구현 |
| * **remote\_agent** | ZeroMQ와 연결하는 여러 agent를 위한 기초적인 클래스 |
| * **seq2seq** | 기본적인 GRU sequence to sequence 모델 |
| * **ir\_baseline** | Information retrieval (IR) 베이스라인 – TFIDF-weighted matching |
| * **repeat\_label** | Merely repeating all data sent to it을 위한 베이직 클래스 (debugging) |

**[6.3 Examples]**

이 디렉토리들은 다른 main들의 예시를 포함하고 있다.

|  |  |
| --- | --- |
| * display\_data | 커맨드 라인에서 제공되는 특정 task로부터 data를 display한다. |
| * display\_model | 제공된 모델의 prediction을 보여준다. |
| * eval\_model | 주어진 task에서의 주어진 모델의 evaluation metrics를 계산한다. |
| * train\_model | 주어진 task와 모델에서 standard training 과정을 실행하고, training과 validation 사이의 가능한 alternating과 logging을 포함한다. |

|  |
| --- |
| 예를 들어, bAbI tasks에서 10개의 random examples를 display할 수 있다.  Python display\_data.py -t babi -n 10  Multitasking bAbI와 SQuAD를 동시에 display할 수 있다.  Python display\_data.py -t babi, squad  Movies Subreddit에서 IR baseline 모델을 평가할 수 있다.  Python eval\_model.py -m ir\_baseline -t ‘#moviedd-reddit’ -dt valid  32개의 example들의 배치 사이즈의 SQuAD 데이터셋 – attentive LSTM 모델을 train할 수 있다.  Python train\_model.py -m drqa -t squad -b 32 |

**[6.4 Tasks]**

첫 출시에 20개 이상의 task들이 지원되었다.

* SQuAD, bAbI tasks, QACNN, QADailyMail, bAbI Dialog tasks, CBT, Ubuntu, VQA – Fig 1

Task들은 5개의 카테고리로 분류된다.

|  |  |
| --- | --- |
| * Question answering (QA) | 다이얼로그의 가장 단순한 형태, 스피커 당 1번의 turn. 모든 지능 다이얼로그 에이전트는 answering questions가 가능해야 한다. 그리고 다양한 종류의 questions들이 있다.  QA는 다른 유형의 다이얼로그보다 평가에 유용하다. |
| * Sentence Completion (Cloze Tests) | Agent가 다이얼로그의 다음 말뭉치에서 missing word를 채워 넣어야 한다.  데이터셋을 만들기 쉽고 평가가 단순하다는 장점이 있다. |
| * Goal-Oriented Dialog | 더 현실적인 task의 클래스, 대화의 끝으로 얻어야 하는 목표가 있다.  예를 들어 손님과 travel agent가 flight에 대해서 얘기하고, 한 스피커가 볼 영화에 대해 추천하고 등등. |
| * Chit-Chat | 명확한 목표를 가지고 있지 않은 대화 task. 그러나 더 많은 discussion. |
| * Visual Dialog | 텍스트뿐만 아니라 이미지에 대한 다이얼로그 |

각 task들은 표준화된 다음의 파일들과 같이 폴더에 포함되어 있다.

|  |  |
| --- | --- |
| * build.py |  |
| * agents.py |  |
| * worlds.py |  |

**[6.5 Mechanical Turk]**

|  |  |
| --- | --- |
| 1. qa\_collector |  |
| 1. model\_evaluator |  |

[7. Demonstrative Experiment]

[8. Related Software]

[9. Conclusion and Outlook]