引き継ぎ資料

黒田佑樹

2022/3/25

1　大まかなディレクトリ構成

研究に用いたプログラム (NH-chat 関連) や論文，外部発表資料等の置き場所を示す．NH_chat と Select_sys_utt に関しては次章以降で説明する．

kuroda/
- 外部発表/
  - IPSJ/ 原稿 (pdf，Tex)，プレゼン資料
  - Sig_Slud/ 原稿 (pdf，Tex)，ポスター
  - 関西合同音声ゼミ/ ポスター
- 研究用プログラム/
  - NH_chat/ユーザとの雑談音声対話を行うプログラム郡
    - ...
  - Select_sys_utt/発話選択戦略の学習と，学習済みモデルを用いたテキスト対話を行うプログラム郡
    - ...
- 卒論/ 論文 (pdf，Tex)，プレゼン資料，発表用アブスト
- 引き継ぎ資料.pdf

2　NH-chat

ユーザとの雑談音声対話を行うプログラム郡．”kuroda/研究用プログラム/NH-chat/README.pdf”に実行環境，実行方法等詳細を記載．

3　Select_sys_utt

強化学習を用いた NH-chat の発話選択戦略の学習と，学習済みモデルを用いたテキスト対話を行うプログラム郡．実行環境は”kuroda/研究用プログラム/Select_sys_utt/Select_sys_utt_env.yml”に記述したものに加えて cuda が使用できれば OK．

3.1　発話選択戦略の学習

NH-chat において，予め用意した 4 つの話題の発話集合 (”kuroda/研究用プログラム/Select_sys_utt/200109_cls8/(CLASSNAME)_bycls_named_(話題名).csv”) 内の発話の選択戦略を強化学習する．具体的には，ある状態 (発話履歴，心象) を入力としたときに，各行動 (発話集合の部分集合) の価値を出力する Q 関数を学習して保存する．

3.1.1 実行コマンド

話題ごとの発話集合に関して Q 関数を学習するためのコマンドの例は以下の通り.

```
python main_RL.py -A train --(オプション)
```

オプションとしては主に以下のようなもの (学習条件やモデル名付与) がある. 詳細は”Select_sys_utt/main_RL.py”内を参照.

【オプション】

`--model str`	学習後に保存する Q テーブル名 (必須)　default: "sample"
`--learning_theme str`	学習対象の発話集合の話題名 (必須)　default: "sports"
	※話題名の候補は現在 ("sports" or "music" or "eating" or "travel")4 つ
`--ep int`	学習エピソード数 (任意)　default: 50000
	※状態数や行動数に応じて増減 (基本は default で ok)
`--m_node int`	DQN の中間層のノード数 (任意)　default: 1000
	※状態数や行動数に応じて増減 (基本は default で ok)
`--seiki bool`	報酬の値を 0〜1 に正規化するかどうか　default: True

基本的にはこれを 4 つの話題の発話集合各々に対して行う. 全ての話題に関して学習を行いたい場合には”kuroda/研究用プログラム/Select_sys_utt/learning.sh”を実行すればよい.

3.1.2 主要プログラムの概要，入出力関係

main_RL.py

agent.py のメソッド learn に引数として，オプションの学習条件を与えて学習を実行. 学習された Q 関数をテーブル化し，メソッド saveQ によって pickle 化して保存.

agent.learn(入力: 学習条件)

指定した発話集合を用いてユーザモデル対システムの対話を行い，各ターンで取得した状態をベクトルに変換しながら，DQN によって学習を行う. 具体的には，状態ベクトルと報酬を与えて，chainerrl.agents.DQN.act_and_train でモデル更新と行動の選択を行う. 報酬は現在の状態と行動を関数 get_reward に入力すると得られる.

chainerrl.agents.DQN.act_and_train(入力: 状態ベクトル，報酬)

現在の状態ベクトルと報酬をもとにモデル更新と行動選択.

agent.get_reward(入力: 状態，行動)

ある状態である行動を取った時の良し悪しを定義しておき，それに従って正負の報酬を出力する.

agent.saveQ(入力: モデル名)

2 次元配列で表現される学習した Q テーブルを，オプション–model で指定した (モデル名) のディレクトリを作成して，その中に (モデル名)_Q という名前で保存.

3.2 学習済みモデルを用いたテキスト対話

学習した，4 つの話題ごとの発話集合の発話選択戦略 (Q テーブル) を用いてテキスト対話を行う. ユーザがテキストで発話と現在の心象 (1〜7) を入力すると，状態に応じたシステム発話が返ってくる. 話題は自動で切り替わり，その都度話題に応じた発話集合と学習済みモデルを用いて発話選択を行う.

3.2.1 実行コマンド

話題ごとの発話集合に関して学習した Q テーブルを用いてテキスト対話を行うためのコマンドは以下の通り.

```
python main_RL.py -A dialogue --(オプション)
```

オプションとしては主に以下のようなもの (学習条件やモデル名付与) がある. 詳細は Select_sys_utt/main_RL.py 内を参照.

【オプション】

```
--model_sports str  スポーツの話題の時の発話選択に用いる Q テーブル (必須)  default: "sample"
--model_music  str  音楽の話題の時の発話選択に用いる Q テーブル (必須)       default: "sample"
--model_eating str  食事の話題の時の発話選択に用いる Q テーブル (必須)       default: "sample"
--model_travel str  旅行の話題の時の発話選択に用いる Q テーブル (必須)       default: "sample"
```

現在最も性能が良い Q テーブルを用いた対話を行うには, "kuroda/研究用プログラム/Select_sys_utt/dialogue.sh" を実行すればよい.

3.2.2 主要プログラムの概要，入出力関係

main_RL.py

agent.py のメソッド conversation に引数として，オプションで指定した Q テーブルを与えて対話を実行. 対話ログ (発話，心象，状態，行動) を，メソッド write_dialogue_log によって csv ファイルとして保存.

agent.conversation(入力: 使用 Q テーブル)

指定した Q テーブルを用いて，ユーザに毎交換発話と心象を入力してもらう形で，ユーザ対システムの対話を行う. 各ターンで取得した状態を入力とし，関数 DialogueEnv.utterance_selection_softmax で価値に沿って行動をサンプリングする. また，ユーザ発話の長さなど対話中の情報を記録しておき，関数 HistoryTheme.decide_Next_Theme に入力することで話題変更の判断を行う.

DialogueEnv.utterance_selection_softmax(入力: 状態)

入力状態における各行動価値を softmax 関数にかけ，各行動の出力確率とする. 出力確率に沿って出力行動をランダムに選択する. また選択行動中に複数発話がある場合，その中からランダムに選択する.

HistoryTheme.decide_Next_Theme(入力: 対話中の情報)

対話中の情報をベクトル化し，学習済み svm モデル (clf_model.pickle) に入力し，話題変更の可否を推論, 出力する.

agent.write_dialogue_log(入力: 対話ログ)

記録していた対話ログ (発話，心象，状態，行動) ををを csv ファイルに書き込む.

3.3 ディレクトリ構造

- Select_sys_utt/
  - 1902themeinfo/ 発話ごとの話題
    - 2105M2001.txt 谷端さんの操作する GUI と同じ形式のサンプルログ
  - 200109_cls8/ 発話集合の情報
    - (CLASSNAME)_freq_cls-theme.csv クラスごと（対話行為ごと）の話題ごとの発話数
    - (CLASSNAME)_average.csv クラスごと（対話行為ごと）の特徴量平均
    - (CLASSNAME)_bycls_named_(話題名).csv 話題ごとの発話集合に状態行動名と blacklist をつけたもの
    - (CLASSNAME)_bycls.csv クラスごとの発話情報
    - (CLASSNAME).csv 使用全交換の特徴量情報
  - refData/
    - 200109_cls8_DAname.csv 対話行為セットにつけた名称
    - 200109_fea7.csv 対話行為セットの作成に用いる特徴量一覧
    - 200110_baseDA4.csv 手で作成した対話行為セット
    - 200110_baseDA4_DAname.csv 手で作成した対話行為セットにつけた名称
    - dependWord.txt 依存単語の一覧
    - exchgUI3Info.csv 発話情報．ユーザモデルに必要
    - nodWord.txt 応答単語の一覧
    - parameters.txt ここで path の管理をしています.
    - reward_bigramDA.csv 報酬の設計に必要
    - simpleDA.csv 報酬で使用した簡単な対話行為
    - thankWord.txt 感謝単語の一覧
    - userID_1902.txt Hazumi1902 で使用する被験者 ID
    - usingTheme.txt 対話での使用話題の指定
  - sample220314_(話題)_*/ 発話集合ごとの強化学習済みモデル (2022/3/25 現在最も性能が高いもの)
    - sample220314_(話題)_*_Q Q テーブル
    - sample220314_(話題)_*_rewards.npy 報酬推移 (numpy 形式)
    - sample220314_(話題)_*_rewards.png 報酬推移 (画像形式)
  - learning.sh 強化学習実行
  - heatmapQ.py 状態行動価値関数を表示する関数
  - dialogue.sh 学習済み Q テーブルを用いたテキスト対話実行
  - main_RL.py メインのコード．強化学習や学習結果を用いた対話を行う.
  - virtual_user.py ユーザモデル
  - dialogue_env.py 対話環境 (状態や行動の設定，発話選択)
  - params.py パラメータや path 管理
  - theme.py 対話話題や発話履歴の管理
  - agent.py 発話選択の強化学習と学習済みモデルを用いた対話
  - clf_model.pickle 話題変更のための svm モデル