NH-chat　README

作成者：奥野 尚己

2021 年 6 月 15 日

1　NH-chat の概要

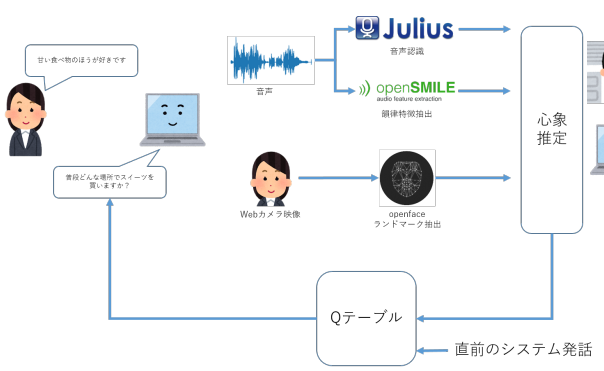

図 1　従来の NH-chat

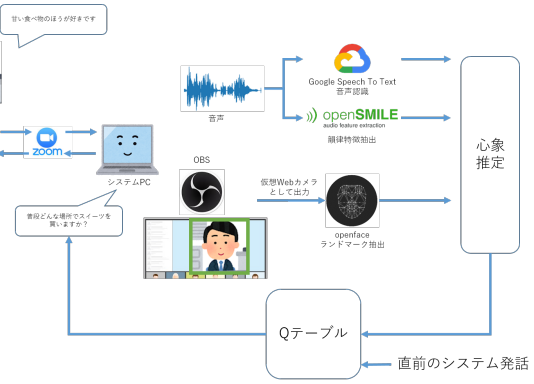

図 2　遠隔 NH-chat

NH-chat とは MMDAgent を用いた音声対話システムである．従来の NH-chat の構成を図 1 に示す．NH-chat の対話におけるシステム発話選択には言語情報，韻律情報，顔画像情報が用いられている．これらの情報を用いて推定されたユーザ心象と過去のシステム発話などを学習済み Q テーブルに入力し，次のシステム発話を選択している．従来の NH-chat はシステムの動作する PC と対面した状態で使用されることを前提としている．ユーザの顔画像は Web カメラの映像を取得することでシステムに入力され，システム発話選択に用いる各情報の取得には以下のアプリが使用されている．

- 言語情報 (音声認識)：Julius
- 韻律情報：openSMILE
- 顔画像情報：OpenFace

遠隔 NH-chat は遠隔かつ自動で対話データを収集することを目的に作成したシステムである．遠隔 NH-chat の構成を図 2 に示す．従来のシステムとの違いは，音声認識に Google Speech To Text を用いる点と，被験者が遠隔で参加する点である．顔画像情報の取得には OBS の仮想 web カ

メラ機能を用いている．具体的には，Zoom で被験者が映るウィンドウをキャプチャし，仮想 web カメラとして出力した映像を OpneFace に入力することで顔画像情報を得ている．

本テキストで導入するシステムは，従来の用途（対面して web カメラを使用する方法）でも遠隔でも使用できる．本テキストではシステムの動作に必要なアプリ，モジュールのインストール方法と，それらのセットアップ方法を示す．具体的な使用方法については実験の手順書を参照．

2 動作環境

開発は windows10 で行った．windows8.1 でも動作確認を行った．アプリのバージョンは以下の通りである．

- MMDAgent 1.7
- MMDAgent Sample Script 1.8
- MMDAgent ソケット通信プラグイン 0.2.2.0
- openSMILE 3.0
- OpenFace 2.2.0
- Google Cloud SDK
- OBS 27.0.0
- OBS VirtualCam 2.0.4
- Voicemeeter Banana 2.0.5.8
- SoX 14.4.2
- dictation-kit(Julius) 4.5

テスト環境では Python3.6 を使用している．テスト環境で使用したモジュール等についての詳細は以下の手順 1 でクローンしたリポジトリ内 NH-chat_env.yml に記載されている．3.2.2 節の手順に従えばテスト環境と同じバージョンのモジュールが一括でインストールできる．

3 導入手順

NH-chat を導入する手順を示す．導入の大まかな手順は以下の通りとなる．

1. git hub からリポジトリをクローン
   プログラムファイルや学習済みモデルなどをダウンロードする
2. conda を用いた仮想環境の構築
   開発者のテスト環境と同等の環境を構築する．
3. 必要なアプリ等のダウンロード
4. モジュールの導入，動作確認

各工程の詳細を以下に示す．

3.1 git hub からリポジトリをクローン

以下のコマンドを実行し git hub からリポジトリをクローンする．

gitclonehttps://github.com/elprimo041/NH-chat.git

フォルダ構造および各ファイル，フォルダの機能は doc/フォルダ構造.pdf 参照．

3.2 conda を用いた仮想環境の構築

開発者環境を再現するために conda を用いる．conda を用いて仮想環境を構築すると，開発環境と同じバージョンの Python，モジュールを一括でインストールすることができる．まず miniconda をダウンロードし，次にクローンしたリポジトリ内にある環境情報が記載されたファイル (setup/NH-chat_env.yml) を使用して開発環境と同等の仮想環境を構築する．

3.2.1 miniconda のインストール

すでに Anaconda や miniconda がインストール済みで conda コマンドが使用できる場合はこの工程をスキップする．

以下のリンクから miniconda をインストールする．

https://docs.conda.io/en/latest/miniconda.html

miniconda インストール時の Python のバージョンは任意のものでよい．ここで選択した Python のバージョンはシステム実行時に使用されるものとは異なるためである．次の工程で開発者と同じバージョン (Python3.6.10) が自動的にダウンロードされる．

コマンドプロンプトで conda コマンドを実行するためにパスを通す．(環境変数 Path に以下を追加) [1)]

path_to_mimiconda3/Scripts

コマンドプロンプトで conda と入力しコマンドが認識されていることを確かめる．

3.2.2 仮想環境の構築

コマンドプロンプトでクローンした NH-chat/setup フォルダに移動する．(NH-chat_env.yml のあるフォルダ) 以下のコマンドを実行し仮想環境を構築する．

conda env create -n NH-chat -f NH-chat_env.yml

システムの実行に必要な Python とモジュールが一括インストールされる．

3.3 必要なアプリ等のダウンロード

NH-chat の動作に必要なアプリのダウンロードを行う．導入するアプリは MMDAgent，openSMILE，OpenFace，Google Cloud SDK，OBS の 5 つである．Julius 音声認識を使用する場合は dictation-kit の導入が追加で必要となる．

---

1) パスの追加方法：https://www.atmarkit.co.jp/ait/articles/1805/11/news035.html

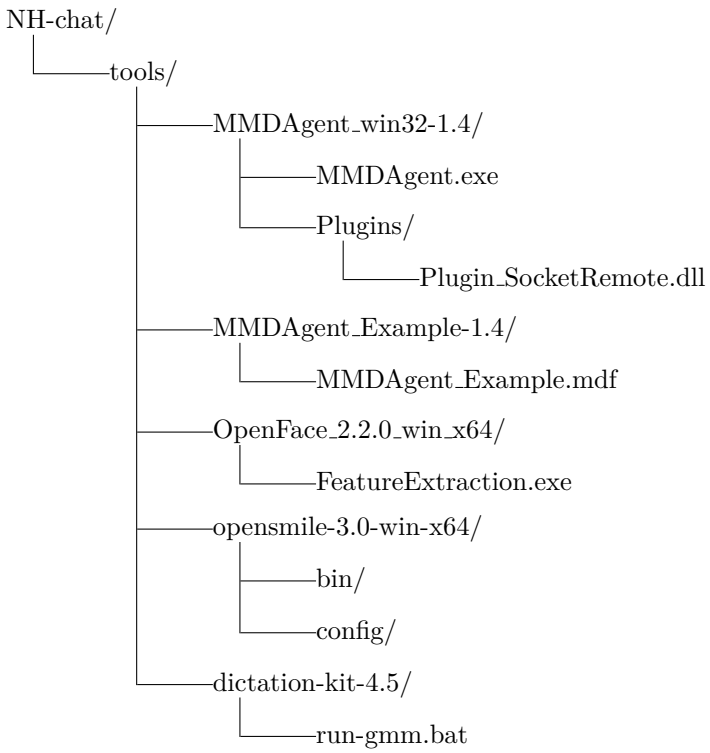

図 3　アプリ導入後のフォルダ構造

MMDAgent，openSMILE，OpenFace はダウンロードした zip ファイルを NH-chat/tools に展開する．展開後のフォルダ構造は図 3 のようになる．

3.3.1　MMDAgent のダウンロード

MMDAgent の本体，サンプルプログラム，ソケット通信用プラグインをダウンロードし，MMDAgent が使用できる環境を構築する．

必要なファイルを以下の手順でダウンロードする．まず下記のリンクから MMDAgent の本体をダウンロードする．

`https://ja.osdn.net/projects/sfnet_mmdagent/downloads/MMDAgent/MMDAgent-1.4/MMDAgent_win32-1.4.zip/`

NH-chat/tools/MMDAgent_Example.mdf をコピーし，NH-chat/tools/MMDAgent_Example-1.4/にコピーする (同名ファイルを置き換える)．

次に下記のリンクから MMDAgent のサンプルプログラムをダウンロードする．

`https://ja.osdn.net/projects/sfnet_mmdagent/downloads/MMDAgent_Example/MMDAgent_Example-1.4/MMDAgent_Example-1.4.zip/`

MMDAgent のソケット通信プラグインを以下のページからダウンロードする．

`https://ux.getuploader.com/cube370/download/15/Plugin\_SocketRemote.zip`

それぞれのファイルを NH-chat/tools に展開する．最後に，ソケット通信を行えるようにするために，MMDAgent 本体の Plugins フォルダに Plugin_SocketRemote.dll をコピーする．

フォルダ構造が図 3 のようになっていることを確かめる．

3.3.2 OpenFace のダウンロード

以下のリンクから OpenFace をダウンロードし NH-chat/tools に展開する．

`https://github.com/TadasBaltrusaitis/OpenFace/releases/download/OpenFace_2.2.0/OpenFace_2.2.0_win_x64.zip`

ダウンロードした OpenFace には顔画像の推定に用いるモデルが付属していない．モデルをダウンロードするために download_model.ps1 を右クリックし Power Shell で実行する．

3.3.3 openSMILE のダウンロード

以下のリンクから opensmile をダウンロードし NH-chat/tools に展開する．

`https://github.com/audeering/opensmile/releases/download/v3.0.0/opensmile-3.0-win-x64.zip`

3.3.4 Google 音声認識の導入

Google 音声認識を使用するには GCP に登録し API キーを含む json ファイルを取得する必要がある．奥野と同じ json ファイルを使用する場合，以下の 1-4 の操作は不要．5 の操作はどちらの場合でも必要となる．奥野とは別のアカウントで Google 音声認識を使用する場合は GCP(Google Cloud Platform) への登録が追加で必要となる．新たなアカウントで json ファイルを取得する手順は以下の通り．

1. Google アカウントを作成し，GCP コンソールにログインしてプロジェクトを作成する
2. Google Speech-to-Text API を有効にする
3.「認証情報」を選択し，「認証情報を作成」から「サービスアカウントキー」をクリックして，json ファイルを作成し，任意の場所に保存する．この json ファイルを読みこむことによって API サービスを使うことができる．
4.
5. 環境変数を以下のように設定する．
   変数名：GOOGLE_APPLICATION_CREDENTIALS
   値：取得した json ファイルの絶対パス

3.3.5 OBS のインストール

下記のリンクから OBS のインストーラをダウンロードする．(開発者環境はバージョン 27.0.0)

`https://github.com/obsproject/obs-studio/releases/`

インストーラを実行し，OBS をインストールする．

OBS で仮想 Web カメラを使うために下記のリンクからプラグインをダウンロードする．

https://obsproject.com/forum/resources/obs-virtualcam.539/ インストーラを実行し，仮想 Web カメラプラグインをインストールする．OBS 27.0.0 ではデフォルトで仮想 Web カメラ機能が搭載されているが，デフォルトの機能ではなぜか OpenFace に入力できない．(黒い画面が読み込まれる？) プラグインで仮想 Web カメラ映像を出力すると OpenFace で読み込める．

3.3.6 SoX のインストール

下記のリンクから SoX のインストーラをダウンロードする．(開発者環境はバージョン 14.4.2)
https://sourceforge.net/projects/sox/files/sox/
インストーラを実行し，SoX をインストールする．コマンドプロンプトで sox コマンドを実行するために環境変数 Path にインストールフォルダを追加する．インストールフォルダはデフォルトでは C:/Program Files (x86)/sox-xx-x-x.

3.3.7 dictation-kit のダウンロード (任意)

Julius 音声認識を利用しない場合はスキップ．
以下のリンクから dictation-kit-4.5 をダウンロードし任意のフォルダに展開する．
https://osdn.net/projects/julius/releases/66544

4 セットアップ

導入したアプリの一部は，対話を行うために以下の項目について準備が必要である．

- 音声入出力の設定
- 組み込み Web カメラと仮想 Web カメラのデバイス番号取得

これらの項目についてセットアップ手順を示す．

4.1 音声入出力の設定

NH-chat では音声ミキサーアプリの VoiceMeeter Banana にすべての音声を入力し，出力を操作している．まず，Windows のサウンド設定を開く．(コントロールパネル→ハードウェアとサウンド→サウンド) 再生タブで以下のように設定する．VoiceMeeter Aux Input:既定の通信デバイス
VoiceMeeter Input:既定のデバイス
録音タブで設定を以下のように変更する．
VoiceMeeter Aux Output:既定の通信デバイス
VoiceMeeter Output:既定のデバイス
ズームでマイクとスピーカーを以下のように設定する．
マイク: VoiceMeeter Output
スピーカー: VoiceMeeter Aux Input

図 4　VoiceMeeter Banana の設定

最後に VoiceMeeter Banana の設定を行う．VoiceMeeter Banana を用いることで，音声認識対象を被験者に限定することができる．

設定後の画面は図 4 のようになる．A が入力を，B が出力を表している．まず，VoiceMeeter Banana を起動し，右上 HARDWARE OUT の A1 でシステムを操作する人が使用する出力機器を指定する．次に Virtual INPUTS の VOICEMEETER VAIO について A1 と B1 を指定する．最後に Virtual INPUTS の VOICEMEETER AUX について A1 と B2 を指定する．

4.2　Web カメラのデバイス番号取得

Web カメラのデバイス番号を確かめるために，以下の手順で入力するデバイスを変更しながら OpenFace を実行する．

1. OBS を起動し出力画面サイズを 960x540 に変更する
   ファイル→設定→映像→基本 (キャンバス) 解像度
2. OBS でトラッキング対象の画面をキャプチャする
   導入テスト時は映像キャプチャデバイスで組み込み web カメラの映像を取り込むとよい
3. 仮想 Web カメラの出力を開始する
   ツール→ VirtualCam → start
   デフォルト機能であるコントロールパネル内仮想カメラ開始ではないので注意
4. コマンドプロンプトで OpenFace の FeatureExtraction.exe があるフォルダ (NH-

chat/tools/OpenFace_2.2.0_win_x64) に移動する

5. 以下のコマンドをデバイスオプションの数字を変更しながら実行する
   FeatureExtraction.exe -cam_width 960 -cam_height 540 -device {DEVICE_NUM}
   開発者環境では，組み込み Web カメラのデバイス番号が 0，仮想 Web カメラが 2
   組み込み Web カメラのデバイス番号を調べる際は OBS の出力を停止する

調べたデバイス番号を NH-chat/src/Interface/tools.py の該当箇所に記載する．
CAMERA_NUM_OFFLINE = { 組み込み Web カメラのデバイス番号 }
CAMERA_NUM_ONLINE = { 仮想 Web カメラのデバイス番号 }

5 モジュールの導入，動作確認

コマンドプロンプトで NH-chat/setup フォルダに移動し，下記のコマンドで仮想環境を起動する．
conda activate NH-chat
Conda 環境が有効になると，行の最初に (NH-chat) が表示される．

intro_confirm.py を実行すると環境構築が正しくできているか確かめられる．MMDAgent は初回起動時，実行許可が必要．

6 プログラムの実行

6.1 実行コマンド

実行前に必要なアプリケーション (zoom，OBS，Voicemeeter Banana) を起動する必要がある．実行コマンドは以下の通りである．

```
cd path-to-NH-chat/NH-chat/src
conda activate NH-chat
python nh-chat.py [options]
```

オンラインモードで起動した場合，コマンド実行後以下の手順で zoom の被験者が映る範囲の画面をキャプチャする必要がある．

1. zoom で MMDAgent を画面共有する
2. 被験者の映る画面を最大化する
3. OBS でソースに画面キャプチャを追加し，被験者の映る範囲を指定する (alt+ドラッグ)
4. 仮想 Web カメラの出力を開始する

6.2 オプション

以下のコマンドを実行することでオプションの一覧を確認することができる．

python nh_chat.py -h

以下の出力が得られる．

```
usage: nh_chat.py [-h] [--model MODEL] [--mode MODE] [-u USER]
                  [--turn_num TURN_NUM] [--text TEXT]
                  [--asr_module ASR_MODULE]
                  [--response_time_no_word RESPONSE_TIME_NO_WORD]
                  [--turn_buffer TURN_BUFFER] [--asr_conf ASR_CONF]
                  [--is_debug IS_DEBUG]

nh_chat.py [Args] [Options] Detailed options -h or --help

optional arguments:
  -h, --help            show this help message and exit
  --model MODEL         強化学習モデル名 (default:200117_c099)
  --mode MODE           実行モード．online か WebCamera(default:WebCamera)
  -u USER, --user USER  ユーザ名 (default:tmp_user)
  --turn_num TURN_NUM   ターン数 (default:3)
  --text TEXT           テキストで対話を行うか (default:False)
  --asr_module ASR_MODULE
                        音声認識モジュール．google か julius(default:google)
  --response_time_no_word RESPONSE_TIME_NO_WORD
                        ユーザターン開始後ユーザが発話しない場合， どの程度待ってターンテイキングを行うか (default:6.0)
  --turn_buffer TURN_BUFFER
                        ユーザ発話の音声認識が確定した後， どの程度待ってターンテイキングを行うか (default:1.5)
  --asr_conf ASR_CONF   初期化時に音声認識テストを行うか (default:False)
  --is_debug IS_DEBUG, --debug IS_DEBUG
                        デバックモードで実行するか (default:False)
```

6.3 実行時のフロー

6.3.1 初期化

nh_chat.NH-chat.__init__で対話のに必要な初期化を行う．init 内では以下の操作が行われる．

- モデルの選択
- ベースタイムの取得
- 強化学習環境の初期化 (RL.dialogue_env.DialogueEnv)
- エージェントの初期化 (RL.agent.TrainedQlearningAgent)
- Q テーブルの読み込み
- テーマの初期化 (..RL.theme.HistoryTheme)
- OpenFace の開始 (Interface.tools.OpenFace.start)

6.3.2 メインループ

nh_chat.NH-chat.run で指定したターン数メインプロセス (nh_chat.NH-chat.process) を回す．nh_chat.NH-chat.process では以下の処理が行われる．

- テーマ変更の判断，テーマ変更 (RL.theme.HistoryTheme.decideNextTheme)
- システム発話生成 (RL.dialogue_env.DialogueEnv.utterance_selection_softmax)
- システム発話 (Interface.tools.MMDAgent.say)
- システム発話ログの記録 (nh_chat.NH-chat.record_log)
- 音声認識 (google_asr.GoogleASR.start)
- 録音開始 (Interface.tools.Record.start)
- ユーザターン終了判定開始 (manage_turn.start_turn_thread)
- ユーザ発話ログの記録 (nh_chat.NH-chat.record_log)
- 韻律特徴抽出 (Interface.tools.OpenSmile.start)
- ユニモーダル心象推定 (nh_chat.NH-chat.predict_*→ RL.predUI.predUnknown)
- マルチモーダル心象推定 (RL.predUI.predUnknown)
- 状態更新 (RL.dialogue_env.DialogueEnv.get_next_state)
- 終了判定

6.3.3 対話終了後

- システム発話ログの保存 (nh_chat.NH-chat.save_log)
- モジュールの終了 (Interface.tools.*.end)

フォルダ構造

- NH-chat/
  - log/ 対話ログ
    - 2105M2001.txt 谷端さんの操作する GUI と同じ形式のサンプルログ
  - setup/ 環境構築時に必要なファイル
    - data 環境構築時のテストで使用するデータ
    - intro_confirm.py 環境構築のテスト
    - NH-chat_env.yml 環境構築時にインストールするモジュール郡
  - src/ ソースコードと設定ファイル
    - Interface 録音，MMDAgent などのインターフェイス部分のコード
    - RL 強化学習部分のコード
    - refData 参照ファイル．パスや Q テーブルなど
  - tools/ インストールする外部アプリ
    - dictation-kit-4.5 Julius
    - MMDAgent_Example-1.4
    - MMDAgent_win32-1.4
    - OpenFace_2.2.0_win_x64
    - opensmile-3.0-win-x64
  - user_data/ 対話時収集したユーザのデータ
    - (ユーザ名)
      - voice/ 録音された音声ファイルと openSMILE の出力
      - (ユーザ名)_face.csv 顔画像情報 (OpenFace の出力)
      - (ユーザ名)_log.txt 対話ログ
      - (ユーザ名)_setting.csv 対話の設定情報