Internship documentation

Multimodal cues of the sense of presence and co-presence in human-virtual agent interaction

J.Bousquet

Table of Contents

1 Introduction 3

1.1 Purpose 3

1.2 References 3

2 Configuration 4

2.1 Machines 4

2.2 Repositories 5

2.3 Versions 5

2.4 Installation 6

3 Corpus 9

4 Versions freeze 12

# Introduction

## Purpose

This documents internship project “Multimodal cues of the sense of presence and co-presence in human-virtual agent interaction“, and specifically the context and corpus, all elements required to reproduce experiments and results, freeze of configuration files, source code, build process (if needed), etc.

The task achieved through this project is to predict sense of presence and co-presence thanks to multimodal cues from a corpus of human-virtual agent interactions, using machine learning techniques (here Random Forests and SVM classifiers). All details regarding the scientific context and experiments can be found in internship report.

# Configuration

## Machines

Three machines were used for development and execution of experiments:

* Personal portable

|  |  |
| --- | --- |
| OS Name | Microsoft Windows 10 Home |
| Version | 10.0.17763 Build 17763 |
| System Manufacturer | ASUSTeK COMPUTER INC. |
| System Type | x64-based PC |
| Processor | Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz, 2301 Mhz, 4 Core(s),  8 Logical Processor(s) |
| Installed Physical Memory (RAM) | 8.00 GB |
| DirectX version: | 12.0 |
| GPU processor: | GeForce GTX 1050 |
| Driver version: | 399.24 |
| CUDA Cores: | 640 |
| Core clock: | 1354 MHz |
| Memory data rate: | 7008 MHz |

* Personal PC

|  |  |
| --- | --- |
| OS Name | Microsoft Windows 10 Home |
| Version |  |
| Motherboard Manufacturer | ASUSTeK COMPUTER INC. |
| System Type | X64-based PC |
| Processor |  |
| Installed Physical Memory (RAM) |  |
| DirectX version: |  |
| GPU processor: | GeForce RTX 1080ti |
| Driver version: |  |
| CUDA Cores: |  |
| Core clock: |  |
| Memory data rate: |  |

* LPL PC

|  |  |
| --- | --- |
| OS Name | Microsoft Windows 10 Professional |
| Version | 10.0.17763 Build 17763 |
| System Type | x64-based PC |
| Processor | Intel Core i7-8700 @ 3.20GHz |
| Installed Physical Memory (RAM) | 32 GB |

## Repositories

Project items are available in following repositories:

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Source** |
| Corpus | A / CorpusHMPassation | <https://lsis-cloud-01.lsis.org/index.php/s/yo50wMOoLnhdmAz> |
| Corpus | B / ACORFORMED | <http://139.124.68.168:5000/> |
| SCM | C / Source code | <https://github.com/jbousque/predictionPresence> |
| SCM | D / Custom SPPAS scripts source code | <https://github.com/gmontcheuil/sppas_scripts> |

## Versions

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Version** | **Type** | **Source** |
| **Python** | 2.7.15 | External tool | <https://www.python.org/downloads/> |
| **SPPAS\_2** | 2.0 | LPL tool | <https://www.ortolang.fr/market/tools>  <http://www.sppas.org/downloads/SPPAS-2.0-2019-01-08.zip> |
| **SPPAS\_1** | 1.8.6 | LPL tool | <http://www.sppas.org/downloads/SPPAS-1.8.6.zip> |
| **MarsaTag** | 0.8.4  2015-04-23 18:12:17 | LPL tool | https://www.ortolang.fr/market/tools/sldr000841 |
| **julius** | 4.3.1 | External tool | <https://fr.osdn.net/projects/julius/downloads/60273/julius-4.3.1-win32bin.zip> |
| **wxPython** | 3.0 | External library | <https://sourceforge.net/projects/wxpython/files/wxPython/3.0.2.0/wxPython3.0-win32-3.0.2.0-py27.exe> |
| **jupyter** | below | External tool |  |
| **pyAudioAnalysis** |  | External library | <https://github.com/tyiannak/pyAudioAnalysis>  (not used anymore) |
| **SPPAS Greg Scripts** | / | Internal library | Repositories / [D] |
| **Corpus** | 1.0 | Internal resource | <http://139.124.68.168:5000/>  (camille / acorformed)  …/ACORFORMED/Data |
| **Corpus (passation)** | 1.0 | Internal resource | <https://lsis-cloud-01.lsis.org/index.php/s/yo50wMOoLnhdmAz?path=%2F> |
| **xlrd** |  | External library | For loading/manipulating excel files |
| **sox** | 14.4.2 |  | <http://sox.sourceforge.net/> |

Main python libraries :

numpy 1.16.1

pandas 0.24.2

scikit-learn 0.20.2

matplotlib 2.2.3

imbalanced-learn 0.4.3

seaborn 0.9.0

xlrd 1.2.0

openpyxl 2.6.0

## Installation

Most tools used are based on Python or Java.

Development environment consists of Python 2 (scripts and jupyter notebooks) and was installed on a PC with Windows 10 64bits (though it should be possible to setup environment on unix based environment).

### Python

Python 2 should be installed alone – anaconda based setup did not work with tools and libraries required. It is normally possible to install both versions of python on same machine (2 and 3) but this was not tested. Source code should normally be compatible with Python 3 (for the notebooks) though Python 2 was used for experiments. Source code for features engineering (most .py files) could not be used with Python 3.

### Python libraries

|  |  |
| --- | --- |
| **Name** | **Install command** |
| **numpy** | pip install numpy |
| **scipy** | pip install scipy |
| **scikit-learn** | pip install scikit-learn |
| **matplotlib** | pip install matplotlib |
| **jupyter notebook** | pip install jupyter |
| **pandas** | pip install pandas |
| **pydub** | pip install pydub |
| **xlrd** | pip install xlrd |
| **openpyxl** | pip install openpyxl |
| **Sox (python wrapper)** | pip install sox |
| **wxPython** | pip install wxPython=3 |
| **unicodecsv** | pip install unicodecsv |
| **sklearn-evaluation** | pip install sklearn-evaluation |
| **imbalanced-learn** | pip install -U imbalanced-learn |

* Install SPPAS
* Prerequisites :
* Install wxPython v3.x (not 4.x !)
* Install julius
* Install python libraries

Note: installing wxPython with conda can break the environment on python 2...

* Install julius
* Donwload and unzip julius-4.3.1-win32bin.zip in folder $JULIUS\_HOME
* Add $JULIUS\_HOME/bin to PATH environment variable
* Copy julius-4.3.1.exe as julius.exe in $JULIUS\_HOME/bin
* Alternatively, copy julius.exe to C:\WINDOWS

Check : open cmd and :

C:\Users\jerem>julius

Julius rev.4.3.1 - based on

JuliusLib rev.4.3.1 (fast) built for i686-pc-cygwin

Copyright (c) 1991-2013 Kawahara Lab., Kyoto University

Copyright (c) 1997-2000 Information-technology Promotion Agency, Japan

Copyright (c) 2000-2005 Shikano Lab., Nara Institute of Science and Technology

Copyright (c) 2005-2013 Julius project team, Nagoya Institute of Technology

Try '-setting' for built-in engine configuration.

Try '-help' for run time options.

* Install SPPAS
* Install MarsaTag

In version used there was a mistake in the windows launcher script of Marsatag, so the part in red below had to be manually added to the last line of the batch file :

|  |  |
| --- | --- |
| MarsaTag-UI.sh | MarsaTag-UI.bat |
| …  $JAVA "${JAVA\_OPTS[@]}" -jar "$JAR" -g "$@" | …  %JAVA% %JAVA\_OPTS% "%ORTO\_OPT%" -jar "%JAR%" -g %\* |

Without this, during POS tagging or other activities involving MarsaTag, the Marsatag UI will show up and expected treatments will not be performed.

* Install pyAudioAnalysis
  + Normally, pyAudioAnalysis is not used anymore and does not have to be installed ! But some .py files (not used during this internship) still reference pyAudioAnalysis.
  + Install dependencies:

pip install numpy matplotlib scipy sklearn hmmlearn simplejson eyed3 pydub

Note: hmmlearn requires Microsoft Visual C++ compiler for python27 : <https://www.microsoft.com/en-us/download/details.aspx?id=44266>

pip command must be executed from 'base' environment in conda !

needs also vc build tools

<https://visualstudio.microsoft.com/downloads/>

If you get error " LINK : fatal error LNK1158: cannot run 'rc.exe'", then copy rc.exe and rcdll.dll from C:\Program Files (x86)\Windows Kits\8.1\bin\x86 to C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin

* + Clone the source of this library:

git clone https://github.com/tyiannak/pyAudioAnalysis.git

* + Install using pip:

pip install -e .

MKL may be required:

pip install -e git+https://github.com/IvanoLauriola/MKLpy#egg=MKLpy

# Corpus

Interactions data files are grouped in folders hierarchy consisting of following items :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Folder** |  | **Description** | **Format** | **Sample Values** |
| **<subject\_identifier>** | | Identifier of subject (medic) | [N|E][1..n][A..F]  Novice or Experimented  Rank (1 or 2 digits)  Condition letter (see below) | "E13A", "E01B", "N1A" … |
| **<condition>** | | Equipment used to interact with subject during session | [Casque|Cave|PC]  (helmet, cave, computer) |  |
| **<subdir>** | |  | [data|Superviseur|Unity|Video] |  |
|  | data | Raw data |  |  |
|  | Superviseur | Chat history |  |  |
|  | Unity | Movement Sensors |  |  |
|  | Video | Video |  |  |

Meaning of condition letter :

|  |  |  |  |
| --- | --- | --- | --- |
| **Letter** | **Cave** | **Casque (Helmet)** | **PC (Computer)** |
| **A** | 1 | 2 | 3 |
| **B** | 1 | 3 | 2 |
| **C** | 2 | 1 | 3 |
| **D** | 3 | 1 | 2 |
| **E** | 2 | 3 | 1 |
| **F** | 3 | 2 | 1 |

Corpus is available in two distinct sets of files with similar folders hierarchy, noted "A" and "B" here.

|  |  |  |
| --- | --- | --- |
| **Corpus** | **Name** | **Source** |
| **A** | CorpusHMPassation | <https://lsis-cloud-01.lsis.org/index.php/s/yo50wMOoLnhdmAz> |
| **B** | ACORFORMED | <http://139.124.68.168:5000/> |

Generally raw data (recorded data) is xml or xra format. Eaf or Textgrid formats are generated by other tools.

Elements in red are used as inputs for computing subject features in feature matrix.

Elements in blue are used as inputs for computing agent features in feature matrix.

(elements in purple are used for both)

« data »

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Corpus** | **Concerns** | **Description** |
| /asr-realtime  /asr-revised  /asr-rev-tag  /asr-tmp  /asr-trans | A | Subject | Various versions of transcriptions in .xra format. Contains tiers :  - ASR-Utterances: the ASR utterance (speech activity) detections, with an index starting from 1  - ASR-Transcription: the ASR (best) transcription  - ASR-Tokens: tokens alignement of the best transcription (produce by the ASR)  - ASR-Alternatives: ASR alternatives transcription  When run in \_real time\_ 4 other tiers are produced:  - ASR-Events: events thrown by the ASR (UtteranceBegin/End, MakeChange, PhraseHypothesis,...)  - ASR-VUMeterState: VUMeter state (Speech/Noise)  - ASR-VUMeterLevel: VUMeter level  - ASR-Hypothesis: the partial recognition (PhraseHypothesis events) |
| **/asr-trans** | A | Subject | Transcription (xra) used as input for features matrix computations (corresponds to asr-revised). |
| /chat-history | A | Subject, Agent | Chat history in xra format. |
| **…-micro.wav** | A | Subject | Recorded HF micro (mainly subject voice) |

« Superviseur » (/session-<date>)

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Corpus** | **Concerns** | **Description** |
| **chat-history.<date>.xml** | B | Subject, Agent | Chat history (reduced format) |
| **000000-00\_micro.wav** | B | Subject | Same as …/data/…-micro.wav from corpus A.  Sometimes only 1 of them is present. |
| <subdirs> | B | Subject, Agent | Various recorded metadata. |

« Unity »

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Corpus** | **Concerns** | **Description** |
| **<nn>.wav** | A, B | Agent | A wav file per IPU of agent voice (<nn> an integer on 1 or 2 digits). |
| **…-Unity-out\_record\_DATE<date>.txt** | B | Subject, Agent | Recorded position sensors values at each time step, in CSV format. |
| CompressData.exe | B |  | Compress unity txt record (removes duplicated lines) |

« Video »

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Corpus** | **Concerns** | **Description** |
| ….mp4 | B | Subject | Recorded video of session |

# Features engineering

There are 2 steps in this project:

* Computation of hand designed features from corpus / features engineering, described in this chapter
* Machine Learning (ML) experiments, described in next chapter

Below are interesting source files for features engineering step that can be found under /src folder (in gray, src files that should belong to this step, but that were not used here):

|  |  |
| --- | --- |
| agent.py | Additional methods for verbal agent features. |
| agentLabels | Defines agent columns names from Unity csv coordinates for entropy agent features. |
| angles.py | To compute angular speed features. |
| config.py | Configuration file, edit to update paths. |
| coordLabels | Defines columns names from Unity csv coordinates for entropy features. |
| diarize.py |  |
| diarizeAllspeech.py |  |
| entropy.py | To compute entropy features (not used in final experiments) |
| **featureExtraction.py** | **MAIN**  **Loops over all samples folders to compute features, based on command-line arguments.** |
| feutils.py | Class FEUtils: various utility methods (paths treatment, metrics computation, …)  Class DataHandler, DatasetHelper, JNCC2Wrapper: see next chapter. |
| ipuseg.py | To compute IPU length feature. |
| listmodules.py |  |
| lookingAround.py |  |
| mdlDiscretizer.py |  |
| pos.py | To compute POS (parts of speech) features, ie ratio1 and ratio2, ie linguistic complexity and lexical richness features. |
| prepro.py |  |
| Read me - code explanation.odt |  |
| subjectLabels | Defines columns names from Unity csv coordinates for subject entropy features. |
| subLabelsFiltered |  |
| test\_datasetHelper.py | Some unit tests. |
| test\_FEUtils.py |
| test\_sppas2.py |
| unity\_display.ipynb | Experiments in jupyter notebook for computation and visualization of rotation data from Unity coordinates. |
| wavSplitter.py | To split wav files into segmented interactions for some verbal features computation. |

## Command-line

Features are extracted from corpus directories through command-line calls to “featureExtraction.py”.

Command --help show the following parameters:

>python featureExtraction.py --help

Logging to C:\IAAA\TER\\_output\logs\predpres-2019-0828\_16-08-42.log

usage: featureExtraction.py [-h] [-s SPLITS [SPLITS ...]] [--agent]

[--candidate CANDIDATE [CANDIDATE ...]]

[--env ENV [ENV ...]] [--pad] [-f F]

optional arguments:

-h, --help show this help message and exit

-s SPLITS [SPLITS ...], --splits SPLITS [SPLITS ...]

list of 3 phases ratio, summing to 1

--agent Whether to compute subject or agent features

--candidate CANDIDATE [CANDIDATE ...]

extract features of specified candidate(s) only

--env ENV [ENV ...] extract features for specified environment(s) only

(same number and order as --candidate should be

provided

--pad Whether to preprocess agent sounds or not

-f F

For example, to generate agent features, with phases defined as 15%, 70% and 15%:

> python featureExtraction.py --agent --splits 0.15 0.70 0.15

Note: any number of phases and ratios can be passed, but only the above configuration was really tested.

To generate agent features, without phases, and with a previous pre-processing on sound files specifically for agent:

> python featureExtraction.py --agent --pad

Note: this pre-processing must mandatorily be done at least once for no phase, and once for phases (for any configuration of phases).

To generate subject (doctor) features, but only for subject E2B in cave environment, and subject E10D in helmet environment:

> python featureExtraction.py --candidate E2B E10D --env PC Casque

## Computing only some features

If you want to restrict the computation loop to, say, IPUlen features, then you can comment out relevant lines from following methods in featureExtraction.py (no command-line argument was added for this purpose):

**def** computeFeatures(pathsList, splitratios, isSubject=True):  
 *# Function to call all functions to compute features* computePOStags(pathsList, splitratios, isSubject)  
 computeSentenceLengths(pathsList, splitratios, isSubject)  
 computeEntropies(pathsList, splitratios, isSubject)  
 removeNaN(splitratios, isSubject)  
 computeIPUlengths(pathsList, splitratios, isSubject)  
 computeAnswerDelays(pathsList, splitratios, isSubject)  
 computeAngularSpeeds(pathsList, splitratios, isSubject)

## Configuration

Configuration in order to localize paths to your environment is all in config.py file, main path to configure is root path (if you follow sub-tree defined in this config.py file for other folders, but of course you can adapt to your case):

ROOT\_PATH = os.path.join(**'C:\\'**, **'IAAA'** ,**'TER'**)

… if your root folder is C:\IAAA\TER.

OUT\_PATH is by default $ROOT\_PATH/\_output, and is also used for the experiments (see next chapter).

# Experiments

Below are interesting source files experiments step that can be found under /src folder (in gray, src files that should belong to this step, but that were not used here):

|  |  |
| --- | --- |
| config.py | Configuration file, edit to update paths. |
| dataset\_nophase.xlsx | An output view of the dataset without phases, generated by machine\_learning.ipynb. |
| dataset\_phases.xlsx | An output view of the dataset with phases, generated by machine\_learning.ipynb. |
| features\_sets.xlsx | Configuration of sets of features, generated/loaded to/from machine\_learning.ipynb. |
| feutils.py | Class FEUtils: see previous chapter.  Class DataHandler: handles experiments paths, saving and loading python objects (pickles) and figures. Relies on config.OUT\_PATH mainly.  Class DatasetHelper: helper methods for target variables discretization.  Class JNCC2Wrapper: A (eventually not used) class to wrap a Java-based implementation of Naïve Credal classifier into scikit-learn API. |
| **machine\_learning.ipynb** | **MAIN**  **Principal jupyter notebook where all experiments were made.** |
| test\_datasetHelper.py | Some unit tests. |
| test\_FEUtils.py |

Notebook **machine\_learning.ipynb** can normally be executed following cells order, in order to reproduce experiments.

Experiments are defined by group, exp\_name, and iteration, which must be configured in a *DataHandler* object.

Group is a root folder that can be used to group experiments together (for example, ‘discretization’).

Exp\_name defines prefix for sub-folder name under group folder, and is supposed to be more specific to the experiment performed (‘predict\_pres\_10folds’).

Iteration can be used if multiple executions of a same experiment (same group and exp\_name) are performed, starting from 0 then increasing it to avoid overwriting previous results.

These params are used to build a sub-tree under config.OUT\_PATH, where all items generated for this experiment will be stored (and can later be loaded from). Scores metrics are normally stored under a sub-folder ‘results’, figures under ‘figures’, and experiments context directly under exp folder.

From the double cross-validation procedure, the *inner* cross\_validation loop is defined in method:

def run\_test:

… and the *outer* cross-validation loop is defined in:

def run\_test\_plan:

Main parameter for *run\_test\_plan* is a python dictionary defining the tests conditions to perform. For each test condition, double cross-validation is performed, and results are stored.

Method *plot\_results* can be used to display the scores as a bar chart graph. A *DataHandler* object is used as param to both *run\_test\_plan* and *plot\_results*, so results can be displayed at any time as long as group, exp\_name and iteration are correct.

With same parameters, *plot\_results\_table* will display exactly the same level of information but as a latex table.

Detailed parameters for all methods should normally be described inline in jupyter notebook cells.

Method *prepare\_train\_data* can be used separately in order to obtain datasets prepared and splitted that can be used out of double cross-validation loops. It handles the splits, centering and reduction (if classifier is not Random Forest), discretization of targets, …

**trash.ipynb** is a dummy notebook where pieces of code from main notebook, were archived when not useful anymore.

END OF DOCUMENT