

SPPAS

the automatic annotation
and analysis of speech



Brigitte Bigi

contact@sppas.org

*Copyright © 2011-2021 – Brigitte Bigi
Laboratoire Parole et Langage – France*

*Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".*

Any and all constructive comments are welcome.

Contents

1	Introduction	1
1.1	About this documentation	1
1.2	What is SPPAS?	1
1.2.1	Overview	1
1.2.2	User engagement	2
1.2.3	Important: about SPPAS 3.+	2
1.2.4	Need help	2
1.2.5	About the author	2
1.2.6	Licenses	3
1.2.7	Supports	4
1.3	Getting and installing	4
1.3.1	Websites	5
1.3.2	Download and install SPPAS	5
1.3.3	The package	5
1.3.4	Update	6
1.4	Features	6
1.4.1	How to use SPPAS?	6
1.4.2	What SPPAS can do?	7
1.5	Main and important recommendations	7
1.5.1	About files	7
1.5.2	About automatic annotations	9
1.5.3	About linguistic resources	9
2	User interfaces	11
2.1	Introduction	11
2.2	Workspaces	11
2.2.1	Overview	11

2.2.2	Create and save a workspace	11
2.2.3	Files	11
2.2.4	References	12
2.2.5	Associate files and references	12
2.3	The graphical user interface	13
2.3.1	Launch the GUI of SPPAS	13
2.3.2	The main frame	13
2.4	The Command-Line user Interface - CLI	14
2.4.1	Usage	15
2.4.2	Arguments for input/output	15
3	Automatic Annotations	17
3.1	Introduction	17
3.1.1	About this chapter	17
3.1.2	THE recommended corpus construction workflow	17
3.1.3	Recordings	19
3.1.4	File formats and tier names	20
3.2	Resources required to annotate	20
3.2.1	Download and install linguistic resources	21
3.2.2	New language support	21
3.3	Annotate with the GUI	21
3.4	Annotate with the CLI	22
3.5	The procedure outcome report	23
3.6	Orthographic Transcription	26
3.7	Search for Inter-Pausal Units (IPUs)	27
3.7.1	Overview	27
3.7.2	The parameters	27
3.7.3	Perform “Search for IPUs” with the GUI	27
3.7.4	Perform “Search for IPUs” with the CLI	29
3.8	Fill in Inter-Pausal Units (IPUs)	30
3.8.1	Overview	30
3.8.2	How does it work	31
3.8.3	Perform “Fill in IPUs” with the GUI	31
3.8.4	Perform “Fill in IPUs” with the CLI	31
3.9	Text normalization	33

3.9.1	Overview	33
3.9.2	Adapt Text normalization	33
3.9.3	Support of a new language	33
3.9.4	Perform Text Normalization with the GUI	34
3.9.5	Perform Text Normalization with the CLI	34
3.10	Phonetization	36
3.10.1	Overview	36
3.10.2	Adapt Phonetization	36
3.10.3	Support of a new language	37
3.10.4	Perform Phonetization with the GUI	37
3.10.5	Perform Phonetization with the CLI	38
3.11	Alignment	39
3.11.1	Overview	39
3.11.2	Adapt Alignment	40
3.11.3	Support of a new language	40
3.11.4	Perform Alignment with the GUI	40
3.11.5	Perform Alignment with the CLI	41
3.12	Activity	42
3.12.1	Overview	42
3.12.2	Perform Activity with the GUI	43
3.12.3	Perform Alignment with the CLI	43
3.13	RMS	43
3.13.1	Overview	43
3.13.2	Perform RMS with the GUI	43
3.13.3	Perform RMS with the CLI	43
3.14	Interval Values Analysis - IVA	44
3.14.1	Overview	44
3.14.2	Perform IVA with the GUI	44
3.14.3	Perform IVA with the CLI	44
3.15	Lexical Metric	44
3.15.1	Overview	44
3.15.2	Perform Lexical Metric with the GUI	45
3.16	Syllabification	45
3.16.1	Overview	45
3.16.2	Adapt Syllabification	45

3.16.3	Support of a new language	46
3.16.4	Perform Syllabification with the GUI	46
3.16.5	Perform Syllabification with the CLI	47
3.17	TGA - Time Groups Analyzer	48
3.17.1	Overview	48
3.17.2	Result of TGA into SPPAS	49
3.17.3	Perform TAG with the GUI	49
3.17.4	Perform TGA with the CLI	50
3.18	Stop words	51
3.19	Self-Repetitions	51
3.19.1	Overview	51
3.19.2	Adapt to a new language	51
3.19.3	Perform Self-Repetitions with the GUI	51
3.19.4	Perform SelfRepetitions with the CLI	52
3.20	Other-Repetitions	52
3.20.1	Overview	53
3.20.2	Adapt to a language and support of a new one	53
3.20.3	Perform Other-Repetitions with the GUI	53
3.20.4	Perform Other-Repetitions with the CLI	53
3.21	Re-Occurrences	54
3.21.1	Perform Re-Occurrences with the GUI	54
3.21.2	Perform Re-Occurrences with the CLI	54
3.22	Momel (modelling melody)	55
3.22.1	Perform Momel with the GUI	56
3.22.2	Perform Momel with the CLI	56
3.23	INTSINT: Encoding of F0 anchor points	57
3.23.1	Perform INTSINT with the GUI	59
3.23.2	Perform INTSINT with the CLI	59
3.24	Face Detection	60
3.24.1	Overview	60
3.24.2	Result of Face Detection	61
3.24.3	Perform Face Detection with the GUI	61
3.24.4	Perform Face Detection with the CLI	61
3.25	Face Identity	63
3.25.1	Overview	63

3.25.2	Perform annotation with the GUI	63
3.25.3	Perform with the CLI	64
3.26	Face Landmarks	64
3.26.1	Overview	64
3.26.2	Perform annotation with the GUI	65
3.26.3	Perform with the CLI	65
3.27	Cued speech - LPC	65
3.27.1	Overview	65
3.27.2	Perform annotation with the GUI	66
3.27.3	Perform with the CLI	66
4	Convert files	67
4.1	Interoperability and compatibility: an introduction	67
4.2	SPPAS conversion method	67
4.3	Supported file formats	68
5	Data Analyses	71
5.1	Introduction	71
5.2	The page Analyze of the GUI	71
5.2.1	Displayed files content	71
5.2.2	The toolbar	72
6	Scripting with Python and SPPAS	81
6.1	Introduction	81
6.2	A gentle introduction to programming	81
6.2.1	Introduction	82
6.2.2	Variables: Assignment and Typing	82
6.2.3	Basic Operators	83
6.2.4	Data types	83
6.2.5	Conditions	85
6.2.6	Loops	86
6.2.7	Dictionaries	86
6.3	Scripting with Python	87
6.3.1	Comments and documentation	87
6.3.2	Getting started with scripting in Python	87
6.3.3	Blocks	88

6.3.4	Functions	89
6.3.5	Reading/Writing files	91
6.3.6	Python tutorials	92
6.3.7	Exercises to practice	93
6.4	anndata, an API to manage annotated data	93
6.4.1	Overview	93
6.4.2	Why developing a new API?	93
6.4.3	The API class diagram	93
6.5	Creating scripts with anndata	96
6.5.1	Preparing the data	96
6.5.2	Read/Write annotated files	96
6.5.3	Manipulating a sppasTranscription object	96
6.5.4	Manipulating a sppasTier object	97
6.5.5	Manipulating a sppasAnnotation object	97
6.5.6	Search in annotations: Filters	98
6.6	More with SPPAS...	101
7	References	103
7.1	References	103
7.1.1	How to cite SPPAS?	103
7.1.2	SPPAS software description	103
7.1.3	About linguistic resources	104
7.1.4	About Search for IPU's	104
7.1.5	About Text Normalization	104
7.1.6	About Phonetization	104
7.1.7	About Forced-Alignment	105
7.1.8	About Syllabification	105
7.1.9	About Repetitions	105
7.1.10	About analyses tools	106
7.1.11	About the API	106
7.1.12	Related references	106
7.2	SPPAS in research projects	108
7.2.1	MULTIPHONIA	108
7.2.2	Amennpro	108
7.2.3	Evalita 2011: Italian phonetization and alignment	108
7.2.4	Cofee: Conversational Feedback	109
7.2.5	Variamu: Variations in Action: a MULtilingual approach	109

8	SPPAS Release notes	111
8.1	The early versions	111
8.1.1	Version 1.0	111
8.1.2	Version 1.1	111
8.1.3	Version 1.2	111
8.1.4	Version 1.3	111
8.2	The birth of SPPAS	112
8.2.1	SPPAS 1.4.0	112
8.2.2	SPPAS 1.4.1	113
8.2.3	SPPAS 1.4.2	114
8.2.4	SPPAS 1.4.3	114
8.2.5	SPPAS 1.4.4	115
8.2.6	SPPAS 1.4.5	115
8.2.7	SPPAS 1.4.6	116
8.2.8	SPPAS 1.4.7	116
8.2.9	SPPAS 1.4.8	117
8.2.10	SPPAS 1.4.9	117
8.2.11	SPPAS 1.5.0	118
8.2.12	SPPAS 1.5.1	118
8.2.13	SPPAS 1.5.2	118
8.2.14	SPPAS 1.5.3	119
8.2.15	SPPAS 1.5.4	119
8.2.16	SPPAS 1.5.5	120
8.2.17	SPPAS 1.5.6	120
8.2.18	SPPAS 1.5.7	120
8.2.19	SPPAS 1.5.8	121
8.2.20	SPPAS 1.5.9	121
8.2.21	SPPAS 1.6.0	122
8.2.22	SPPAS 1.6.1	122
8.2.23	SPPAS 1.6.2	122
8.2.24	SPPAS 1.6.3	123
8.3	The gebinnings of SPPAS	123
8.3.1	SPPAS 1.6.4	123
8.3.2	SPPAS 1.6.5	124
8.3.3	SPPAS 1.6.6	124

8.3.4	SPPAS-1.6.7	126
8.3.5	SPPAS-1.6.8	126
8.4	The development phase	127
8.4.1	SPPAS-1.6.9	127
8.4.2	SPPAS-1.7.0	128
8.4.3	SPPAS-1.7.1	128
8.4.4	SPPAS-1.7.2	129
8.4.5	SPPAS-1.7.3	129
8.4.6	SPPAS-1.7.4	130
8.4.7	SPPAS-1.7.5	130
8.4.8	SPPAS-1.7.6	131
8.4.9	SPPAS-1.7.7	131
8.4.10	SPPAS-1.7.8	132
8.4.11	SPPAS-1.7.9	132
8.5	The stabilization phase	133
8.5.1	SPPAS-1.8.0	133
8.5.2	SPPAS-1.8.1	133
8.5.3	SPPAS-1.8.2	134
8.5.4	SPPAS-1.8.3	134
8.5.5	SPPAS 1.8.4	134
8.5.6	SPPAS 1.8.5	135
8.5.7	SPPAS 1.8.6	135
8.5.8	SPPAS 1.9.0	135
8.5.9	SPPAS 1.9.1	136
8.5.10	SPPAS 1.9.2	137
8.5.11	SPPAS 1.9.3	137
8.5.12	SPPAS 1.9.4	137
8.5.13	SPPAS 1.9.5	138
8.5.14	SPPAS 1.9.6	138
8.5.15	SPPAS 1.9.7	138
8.5.16	SPPAS 1.9.8	139
8.5.17	SPPAS 1.9.9	140
8.5.18	SPPAS 2.0	141
8.5.19	SPPAS 2.1	141
8.5.20	SPPAS 2.2	142

8.5.21	SPPAS 2.3	142
8.5.22	SPPAS 2.4	143
8.5.23	SPPAS 2.5	143
8.5.24	SPPAS 2.6	144
8.5.25	SPPAS 2.7	144
8.5.26	SPPAS 2.8	145
8.5.27	SPPAS 2.9	145
8.5.28	Development	145
8.5.29	Resources	145
8.5.30	Annotations	145
8.5.31	Known bugs:	145
8.6	Migrate to Python 3	146
8.6.1	SPPAS-3.0	146
8.6.2	SPPAS-3.1	146
8.6.3	SPPAS-3.2	147
8.6.4	SPPAS-3.3	148
8.6.5	SPPAS-3.4	148
8.6.6	SPPAS-3.5	149
8.6.7	SPPAS-3.6	150
8.6.8	SPPAS-3.7	150
8.6.9	SPPAS-3.8	151
8.6.10	SPPAS-3.9	152
8.6.11	Next...	153
9	Appendix	155
9.1	List of error messages	155
9.1.1	Global errors	155
9.1.2	From “annotations” package	155
9.1.3	From “utils” package	156
9.1.4	From “audiodata” package	156
9.1.5	From “calculus” package	157
9.1.6	From “plugins” package	157
9.1.7	From “resources” package	157
9.1.8	From “structs” package	158
9.1.9	From “models” package	158
9.2	GNU Free Documentation License	158

Introduction

1.1 About this documentation

This documentation is governed by [GNU Free Documentation License, version 1.3](#). It will assume that you are using a relatively recent version of SPPAS. There's no reason not to download the latest version whenever released: it's easy and fast!

Any and all constructive comments are welcome.

1.2 What is SPPAS?

1.2.1 Overview

SPPAS - the automatic annotation and analyses of speech is a scientific computer software package. SPPAS is daily developed with the aim to provide a robust and reliable software. Available for free, with open source code, there is simply no other package for linguists to simple use in both the automatic annotations of speech and the analyses of any kind of annotated data. You can imagine the annotations or analyses you need, SPPAS does the rest! It doesn't do? Send your suggestion to the author!

Annotating recordings is very labor-intensive and cost-ineffective since it has to be performed manually by experienced researchers with many hours of work. As the primary functionality, SPPAS proposes a set of **automatic or semi-automatic annotations of recordings**. In the present context, annotations are “defined as the practice of adding interpretative, linguistic information to an electronic corpus of spoken and/or written language data. ‘Annotation’ can also refer to the end-product of this process” (Leech, 1997). SPPAS automatizes the annotation processes and allows users to save time. In order to be used efficiently, SPPAS expects a rigorous methodology to collect data and to prepare them.

Linguistics annotation, especially when dealing with multiple domains, makes use of different tools within a given project. This implies a rigorous annotation framework to ensure compatibilities between annotations and time-saving. SPPAS annotation files are in a specific XML format with extension `xra`. **Annotations can be imported from and exported to a variety of other formats** including Praat (TextGrid, PitchTier, IntensityTier), Elan (eaf), Transcriber (trs), Annotation Pro (antx), Phonedit (mrk), Sclite (ctm, stm), HTK (lab, mlf), subtitles formats (srt, sub), CSV files...

“[...] when multiple annotations are integrated into a single data set, inter-relationships between the annotations can be explored both qualitatively (by using database queries that combine levels) and quantitatively (by running statistical analyses or machine learning algorithms)” (Chiarcos 2008). As a consequence, the annotations must be time-synchronized: annotations need to be time-aligned in order to be useful for purposes such as analyses. Some special features are offered in SPPAS for **managing annotated files and analyzing data**. Among others, it includes a tool to filter multi-levels annotations (Bigi and Saubesty, 2015). Other included tools are to estimate descriptive statistics, a version of the Time Group Analyzer (Gibbon 2013), manage annotated files, manage audio files, etc. These data analysis tools of SPPAS are mainly proposed in the Graphical User Interface. However, advanced users can also access directly the Application Programming Interface, for example to estimate statistics or to manipulate annotated data.

1.2.2 User engagement

By using SPPAS, **you agree to cite a reference in your publications**. The full list of references is available in Chapter 7.

1.2.3 Important: about SPPAS 3.+

Python 2.7 reached the end of its life.

For a full support of SPPAS 3.+, upgrade your Python to 3.x as Python 2.7 is no longer maintained, i.e. no new bug reports, fixes, or changes will be made to SPPAS based on Python 2.

1.2.4 Need help

1. Many problems can be solved by updating the version of SPPAS.
2. When looking for more detail about some subject, one can search this documentation. This documentation is available in-line - see the SPPAS website, it is also included in the package in PDF format.
3. There is a F.A.Q. in the SPPAS web site.
4. There are tutorials in the SPPAS web site.
5. If none of the above helps, you may contact the author by e-mail. Do not expect an answer if you don't indicate clearly:
 1. your operating system and its version,
 2. the version of SPPAS (supposed to be the last one), and
 3. the log file,
 4. for automatic annotations, send the report file, and a sample of the data on which a problem occurs.

1.2.5 About the author

Since January 2011, **Brigitte Bigi** is the main author of SPPAS. She has a tenured position of researcher at the French CNRS - [Centre National de la Recherche Scientifique](#). She's working since 2009 at Laboratoire Parole et Langage in Aix-en-Provence, France.

More about the author:

- <http://www.lpl-aix.fr/~bigi>
- <https://orcid.org/0000-0003-1834-6918>

Contact the author by e-mail:

- to improve the quality of the linguistic resources;
- to add new linguistic resources;
- to help in development;
- to add new annotations or analysis methods;
- to declare an issue;
- or to send any other constructive comment!

Do not contact the author if:

- you failed to install Python or wxPython;
- you didn't followed the tutorials;
- you didn't read the documentation.

Possible e-mails are:

- *contact@sppas.org* for general purposes;
- *develop@sppas.org* for developer questions or for a bug alert.

Contributors

Here is the list of other contributors in programming:

- April 2012-June 2012: Alexandre Ranson (1st website);
- April 2012-July 2012: Cazembé Henry (1st GUI);
- April 2012-June 2013: Bastien Herbaut (help system);
- March 2013-March 2014: Tatsuya Watanabe;
- April 2015-June 2015: Nicolas Chazeau (audio support);
- April 2015-June 2015: Jibril Saffi (annotated data support)
- April 2019-June 2019: Barthélémy Drabczuk (workspaces and num2letter)
- April 2020-June 2020: Laurent Vouriot (workspaces and SPEAKER annotations)
- April 2020-June 2020: Florian Hocquet (installer and video support)

1.2.6 Licenses

SPPAS software, except documentation and resources, are distributed under the terms of the [GNU GENERAL PUBLIC LICENSE v3](#).

Linguistic resources of SPPAS are either distributed:

- under the terms of the “GNU GENERAL PUBLIC LICENSE, v3”, or
- on the terms of the “Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License”.

See the chapter 4 of this documentation for details about individual license of the proposed resources.

To summarize, SPPAS users are:

- free to study the source code and the resources of the software they use,
- free to share the software and resources with other people,
- free to modify the software and resources.

1.2.7 Supports

2011-2012:

Partly supported by ANR OTIM project (Ref. Nr. ANR-08-BLAN-0239), Tools for Multimodal Information Processing. Read more at: <http://www.lpl-aix.fr/~otim/>

2013-2015:

Partly supported by ORTOLANG (Ref. Nr. ANR-11-EQPX-0032) funded by the « Investissements d’Avenir » French Government program managed by the French National Research Agency (ANR). Read more at: <http://www.ortolang.fr/>

2014-2016:

SPPAS was also partly carried out thanks to the support of the following projects or groups:

- CoFee - Conversational Feedback <http://cofee.hypotheses.org>
- Variamu - Variations in Action: a MULTilingual approach <http://variamu.hypotheses.org>
- Team C3i of LPL <http://www.lpl-aix.fr/~c3i>
- Campus France, Procore PHC.

2017-2020:

The introduction of Naija language is supported by the ANR [NaijaSynCor](#).

2019-2020:

The introduction of workspaces to manage files and the SPEAKER annotation type were both supported by the Vapvisio ANR project (ANR-18-CE28-0011-01).

1.3 Getting and installing

A tutorial is dedicated to the download and installation of SPPAS.

1.3.1 Websites

The main website of SPPAS is located at the following URL:

<http://www.sppas.org>

Click on “Get it -> Download” to get access to the last release and to some of the recent ones.

The source code is hosted by sourceforge at:

<https://sourceforge.net/projects/sppas/>

1.3.2 Download and install SPPAS

The main website contains the `Download` page to download recent versions, and an installation guide is also available: <http://www.sppas.org/installation.html>. Moreover, a tutorial is describing each of the download and installation steps (series 2). The tutorial is in French but English subtitles are available.

To summarize:

- STEP 1: install Python 3+
- STEP 2: download and decompress the SPPAS package. There is a unique version of SPPAS which does not depend on the operating system. SPPAS is ready to run, so it does not need elaborate installation. All you need to do is to copy the SPPAS package from the website to somewhere on your computer. Choose *a location with preferably only US-ASCII characters in the full name of the path*. The package of SPPAS is compressed and zipped, so you will need to *decompress and unpack* it once you’ve got it.
- STEP 3: launch the `setup.bat` (Windows) or `setup.command` (linux, macos) to finish the installation in GUI mode or use the program `sppas\bin\preinstall.py` to do the same with the CLI. It will install external programs to enable some features - including the GUI, and the linguistic resources.

Notice that administrator rights are required to perform steps 2 and 3.

In case of difficulty arising from this setup, you’re invited to consult the web first. It probably will provide the solution. If, however, the problems were to persist, contact the author by e-mail.

1.3.3 The package

Unlike many other software tool, SPPAS is not distributed as an executable program only. Instead, **everything is done so that users can check / change operation**. It is particularly suitable for automatic annotations: it allows anyone to adapt automatic annotations to its own needs. The package of SPPAS is then a directory with content as files and folders.

The SPPAS package contains:

- the `README.txt` file, which aims to be read
- the files `setup.bat` and `setup.command` to install some external programs
- the files `sppas.bat` and `sppas.command` to launch the Graphical User Interface

- the `resources` directory contains data that are used by automatic annotations (lexicons, dictionaries, ...)
- the `samples` directory contains data of various languages; they are distributed to test various features of SPPAS
- the `plugins` directory
- the `sppas` directory contains the program itself
- the `documentation` directory contains:
 - the terms of the licenses
 - the printable documentation
 - the printable list of features
 - the printable version of the main reference published in “the Phonetician” journal
 - the orthographic transcription convention
 - the folder `scripting_solutions` is a set of python scripts corresponding to the exercises proposed in the chapter “Scripting with Python and SPPAS”

1.3.4 Update

SPPAS is constantly being improved and new packages are published frequently (about 10 versions a year). It is important to update regularly in order to get the latest features and corrections.

Updating SPPAS is very easy and fast: apply steps 2 and 3 of the installation process.

1.4 Features

1.4.1 How to use SPPAS?

There are three main ways to use SPPAS:

1. The Graphical User Interface (GUI) is as user-friendly as possible:
 - double-click on the `sppas.bat` file, under Windows;
 - double-click on the `sppas.command` file, under MacOS or Linux.
2. The Command-line User Interface (CLI), with a set of programs, each one essentially independent of the others, that can be run on its own at the level of the shell.
3. Scripting with Python and SPPAS provides the more powerful way.

Features of SPPAS can then be used either with a Command-line User Interface (CLI) or a Graphical User Interface (GUI). This latter requires wxPython to be installed but not the former. So, there’s no specific difficulty by using this software.

Advanced users can also access directly the Application Programming Interface - API.

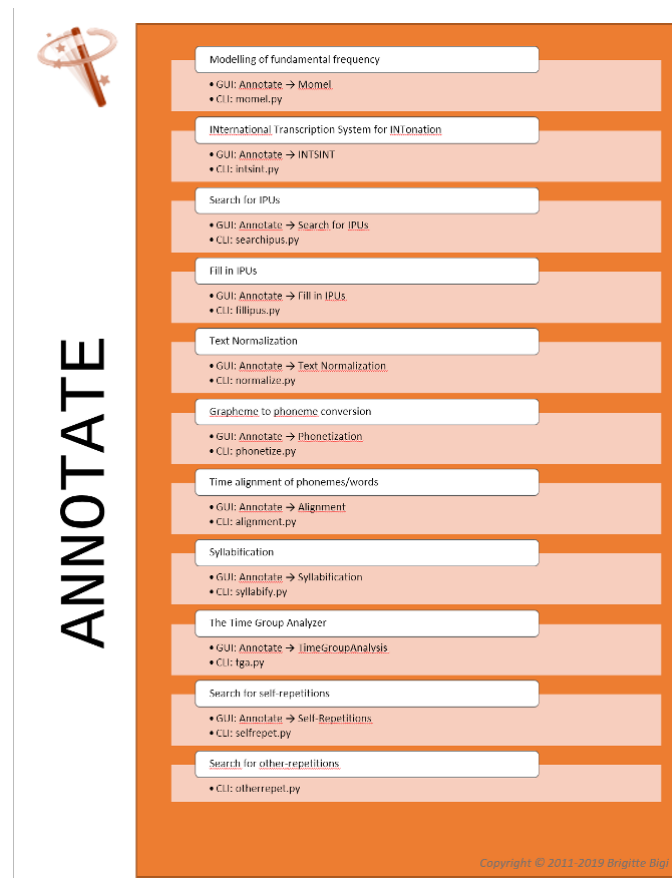


Figure 1.1: SPPAS Automatic annotations

1.4.2 What SPPAS can do?

Features of SPPAS can be divided into 3 main categories:

1. Annotate
2. Analyze
3. Convert

The three next figures list the features of each category and the interface to get access to it.

1.5 Main and important recommendations

1.5.1 About files

There is a list of important things to keep in mind while annotating with SPPAS. They are summarized as follows and detailed in the chapters of this documentation:

1. Speech audio files for automatic annotations:
 - only `wav` and `au` files are supported

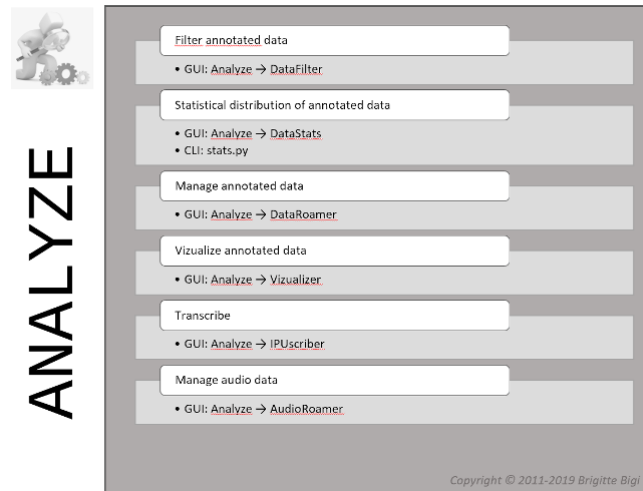


Figure 1.2: SPPAS Automatic analysis

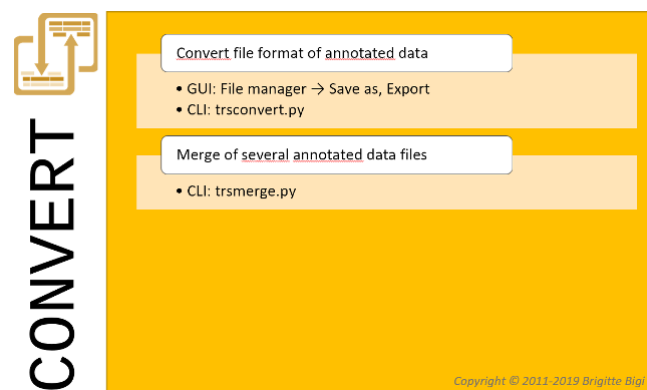


Figure 1.3: SPPAS Automatic file conversion

- only mono (= one channel) files are supported
 - frame rate is *preferably* 16000hz
 - bit rate is *preferably* 16 bits
 - good recording quality is expected. It is obviously required to never convert from a compressed file, like mp3 or aac for example.
2. Annotated data files:
 - *UTF-8* encoding only
 3. It is recommended to use only US-ASCII characters in file names (obviously it includes its path)

1.5.2 About automatic annotations

The quality of the results for most of the automatic annotations is highly influenced by **the quality of the data the annotation takes in input**. This is a politically correct way to say: *Garbage in, garbage out!*

Annotations are based on the use of linguistic resources. Resources for several languages are gently shared and freely available in the package of SPPAS. The quality of the automatic annotations is largely influenced by **the quality of the linguistic resources**. Any help is welcome to improve or add resources.

1.5.3 About linguistic resources

Users are of crucial importance for resource development.

The users of SPPAS are invited to contribute to improve them. They can release the improvements to the public, so that the whole community benefits.

User interfaces

2.1 Introduction

The current chapter describes how to manage workspaces, i.e a set of files and thier description. Then it describes how to use the Graphical User Interface (GUI) and the Command-line User Interface (CLI).

2.2 Workspaces

2.2.1 Overview

Since version 2.3, a workspace manager was introduced into SPPAS. It allows to manage the files and the references. All the automatic annotations are using the workspaces.

2.2.2 Create and save a workspace

A workspace is described in a configuration file into the folder “workspaces” of the SPPAS package. This file is made of the name of the workspace followed by the extension “.json”. It includes all properties of a workspace that are described in this chapter.

This configuration file can be copied into any other directory and re-imported into SPPAS.

2.2.3 Files

A workspace can manage a set of files. Each time a file is added into the workspace, its “root” and its “path” are extracted and stored into a tree-like architecture. Data are strutured as follow:

- a workspace contains a list of paths,
- each path contains the list of roots sharing this path,
- each root contains the list of file names sharing this root.

For example, if the file “/sppas/samples/samples-eng/oriana1.wav” is added into the workspace, the following next actions will be performed:

- the path “/sppas/samples/samples-eng/” is added,
- the root “/sppas/samples/samples-eng/oriana1” is added into the path,
- the filename “/sppas/samples/samples-eng/oriana1.wav” is added into the root. Several properties of the file are also stored like its size and the date of modification.

Then, when an annotated file is created, its filename is inserted into the already existing root.

The annotation manager is automatically searching for a given input pattern in the filenames of a given root and is creating a file with a different output pattern. Notice that a pattern:

- must start by the character “-”;
- must contain at least 2 characters.

Thanks to this new management of files, the input and output patterns of the annotations can be modified. The second advantage of such management of files into a workspace is to add all files of a corpus only once into a workspace and to save it so that it is ready-to-use each time SPPAS is started.

To summarize, if we consider the file “/sppas/samples/samples-eng/oriana1-token.TextGrid”, we have:

1. “/sppas/samples/samples-eng/” is the *path*
2. “/sppas/samples/samples-eng/oriana1” is the *root*
3. “/sppas/samples/samples-eng/oriana1-token.TextGrid” is the *filename*, in which: - “-token” is the *pattern*, - “.TextGrid” is the *extension*.

2.2.4 References

There are 3 different types of references: STANDALONE, SPEAKER and INTERACTION. Each of them corresponds to an annotation type. The annotations of both SPEAKER and INTERACTION require to associate roots with these references.

Each reference can contain a list of values with an identifier key to define them. Here is an example of a reference “John” of type SPEAKER with its list of values:

- firstname: John (str)
- gender: male (str)
- age: 45 (int)
- L1: eng (str)
- L2: fra (str)

It is recommended to use only us-ascii characters and no whitespace for the key.

2.2.5 Associate files and references

Roots and references should be associated. In the previous example, it could allow to declare the list of files related to a given speaker and the same to declare interactions.

The definition of such references/values, and their link to the corresponding roots allow to perform an elaborated way to check files of a workspace. It could be easy for example to check all files with file extension .wav of “male” gender more than 40 years old.

2.3 The graphical user interface

2.3.1 Launch the GUI of SPPAS

Both the main windows and a “Log window” will open. The main frame is made of a menubar at top, a main content in the middle and buttons for actions at bottom.

Most of the existing/incoming tutorials are explaining how to access features of SPPAS with this GUI. This chapter only summarizes some of them.

Under Windows

Once the SPPAS package is opened in the File Explorer, double-click on the `sppas.bat` file.

In recent versions of Windows (e.g. 10), the first time you try to run SPPAS you may get a window with title “Windows protected your PC” and the following message: “Windows SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk. More info”. Click `More info` message and then `Run anyway` button. The file will now run SPPAS, and you will now no longer get a Windows protected your PC prompt when you run this specific file next time. This warning message comes from the fact that SPPAS is a free software and we did not paid to Microsoft commercial fees which would remove the “Unknown Publisher” warnings.

Under MacOS

Once the SPPAS package is opened in the Finder, double-click on the `sppas.command` file.

The first time you try to run SPPAS you may get a message: “sppas.command can’t be opened because it is from an unidentified developer.”. This warning message comes from the fact that SPPAS is a free software and we did not paid to Apple commercial fees. The solution is to run SPPAS with a right click (alt-click) on `sppas.command` file. This time you will get a message: “sppas.command is from an unidentified developer. Are you sure you want to open it?” Then click on `Open`. It will also now work each time you run it.

Under Linux

Once the SPPAS package is opened in the File Explorer, double-click on the `sppas.command` file. Either, open a terminal and run it. Last solution is to directly execute `sppas/bin/sppasgui.py`.

2.3.2 The main frame

The top menu allows to open various pages like “Files”, “Annotate” or “Analyze”. To browse through the pages, use the mouse to click its button in the menubar or use keyboard shortcuts. Under Windows, press `CTRL` and under MacOS, press `COMMAND`. Then, click left-right arrows to navigate to the previous/next page. Use up-arrow to go to the 1st page and Down-arrow to the last page. Use “`CTRL+F`”/“`COMMAND+F`” to go to the “Files” page.

To change colors and fonts, click on the `Settings` icon at bottom. These settings are saved in a file to be used each time SPPAS is executed.

The `About` action button allows to display the main information about SPPAS: author, license, a link to the web site, etc.

Analyze page

It allows to display a summary of each file: either audio or annotated files can be opened and analyzed. The following analyses can be performed:

- manage tiers: duplicate, remove, copy, move, ...
- estimate distribution statistics on tiers;
- create new tiers with the filter systems, either the “simple” or the “relation one”;
- check if an audio file is compatible with the automatic annotations.

Edit page

It allows to annotate manually and to view files in a timeline. It can play several media at a time, it displays the list of annotations of the opened tiers...

Plugins page

Installing plugins is a very useful solution to extend the features of SPPAS. Several plugins are available for download in the main site of SPPAS. The plugins of SPPAS are installed in a folder with name “plugins” in the main directory of SPPAS.

To install a new plugin, simply follow this workflow:

1. Create or download the plugin package - e.g. a zip file.
2. Execute SPPAS.
3. Click on the ‘Plugin’ icon then click on the ‘Install’ button of the toolbar.
4. Browse to indicate the plugin package.
5. See the new plugin icon in the plugins list.

To delete a plugin, click on the “Delete” button of the toolbar. Choose the plugin in the given list then click on the “OK” button. Notice that the plugin is definitively deleted of the disk.

To execute a plug-in, select file(s) in the File explorer, click on the icon of the plug-in and follow instructions of the plugged program.

2.4 The Command-Line user Interface - CLI

A command-line user interface (CLI) is a means of interacting with a computer program where the user issues commands to the program in the form of successive lines of text (command lines). Command-line interfaces provide a more concise and powerful means to control the program than the GUI.

Operating system command line interfaces are called a command-line interpreter, command processor or shell. It displays a prompt, accept a “command line” typed by the user terminated by the Enter key, then execute the specified command and provide textual display of results or error messages. When a shell is active a program is typically invoked by typing its name followed by command-line arguments (if any).

Such programs are located in the `bin` folder of the `sppas` directory of the package. All these programs are written with the programming language Python and are both compatible with version 2.7 and 3.4+. They do not use a GUI so that installing wxPython is not required. The `alignment.py` program also requires Julius or HVite.

2.4.1 Usage

It is usual for a program to be able to display a brief summary of its parameters. Each program included in SPPAS provides its usage by using the option `--help`, as for example:

```
prompt> python .\sppas\bin\trsconvert.py --help
usage: trsconvert.py [files] [options]

... a program to export annotated files.

optional arguments:
  -h, --help      show this help message and exit
  --quiet         Disable the verbosity
  --debug         Highest level of verbosity

Files:
  -i file         Input annotated file name.
  -o file         Output annotated file name.

Options:
  -n value        Number of a tier (use as many -n options as wanted). Positive
                  or negative value: 1=first tier, -1=last tier.
  -t tiername      Name of a tier (use as many -t options as wanted).
```

This program is part of SPPAS version 2.0. Copyright (C) 2011-2019 Brigitte Bigi. Contact the author at: contact@sppas.org

2.4.2 Arguments for input/output

In most of the programs, there is an option `-i` for the input file. There's no specific constraint on this file name. For example, the following program will execute the Momel automatic annotation:

```
python .\sppas\bin\momel.py -i .\samples\samples-eng\ENG_M15_ENG_T02.PitchTier
```

With this option `-i`, a name of an output file can be given with the option `-o`; if not, the main part of the result is printed on the standard output.

In several programs, an option `-I` can also be available to execute the program, and several files can be processed with this option. Moreover, there is some flexibility in file names with this option. SPPAS will search for the appropriate file from the given file name. For example, the next commands will process the same:

```
python .\sppas\bin\intsint.py -I .\samples\samples-eng\ENG_M15_ENG_T02.wav
python .\sppas\bin\intsint.py -I .\samples\samples-eng\ENG_M15_ENG_T02.PitchTier
python .\sppas\bin\intsint.py -I .\samples\samples-eng\ENG_M15_ENG_T02-momel.xra
```

With the option `-I`, the name of the output file is fixed and can't be changed. For example, the previous example will create a file with name `.\samples\samples-eng\ENG_M15_ENG_T02-intsint.xra`. An option `-e` allows to choose the extension of the file, `.xra` is the default one.

The options to manage input/output files can be summarized as follow:

Files (manual mode):

-i file	An input file.
-o file	Output file name (optionnal).

Files (auto mode):

-I file	Input file (append).
-e .ext	Output file extension. One of: .xra .TextGrid .eaf .csv .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff

Automatic Annotations

3.1 Introduction

3.1.1 About this chapter

This chapter describes the followings:

- what is the goal of the annotation;
- what are the requirements (files, tiers);
- what kind of resources are needed;
- what is the expected result;
- how to perform it within SPPAS.

However, this chapter is not a description on how each automatic annotation is working and what to expect about it: there are references for that in chapter 8.

Among others, SPPAS is able to produce automatically annotations from a recorded speech sound and its orthographic transcription. Let us first introduce what is **the recommended way to annotate a corpus with SPPAS**.

3.1.2 THE recommended corpus construction workflow

The kind of process to implement in the perspective of obtaining rich and broad-coverage multimodal/multi-levels annotations of a corpus is illustrated in next Figure. It describes each step of the annotation workflow. Obviously, there are other ways to construct a corpus but 1/ do not blame SPPAS if you don't get the results you expected, and 2/ do not contact the author if you have chosen to not follow these recommendations.

This Figure must be read from top to bottom and from left to right, starting by the recordings and ending to the analysis of annotated files. Yellow boxes represent manual annotations, blue boxes represent automatic ones.

After the recording an audio file (see recordings recommendations), the first annotation to perform is to search for the IPUs. Indeed, at a first stage, the audio signal must be automatically segmented into Inter-Pausal Units

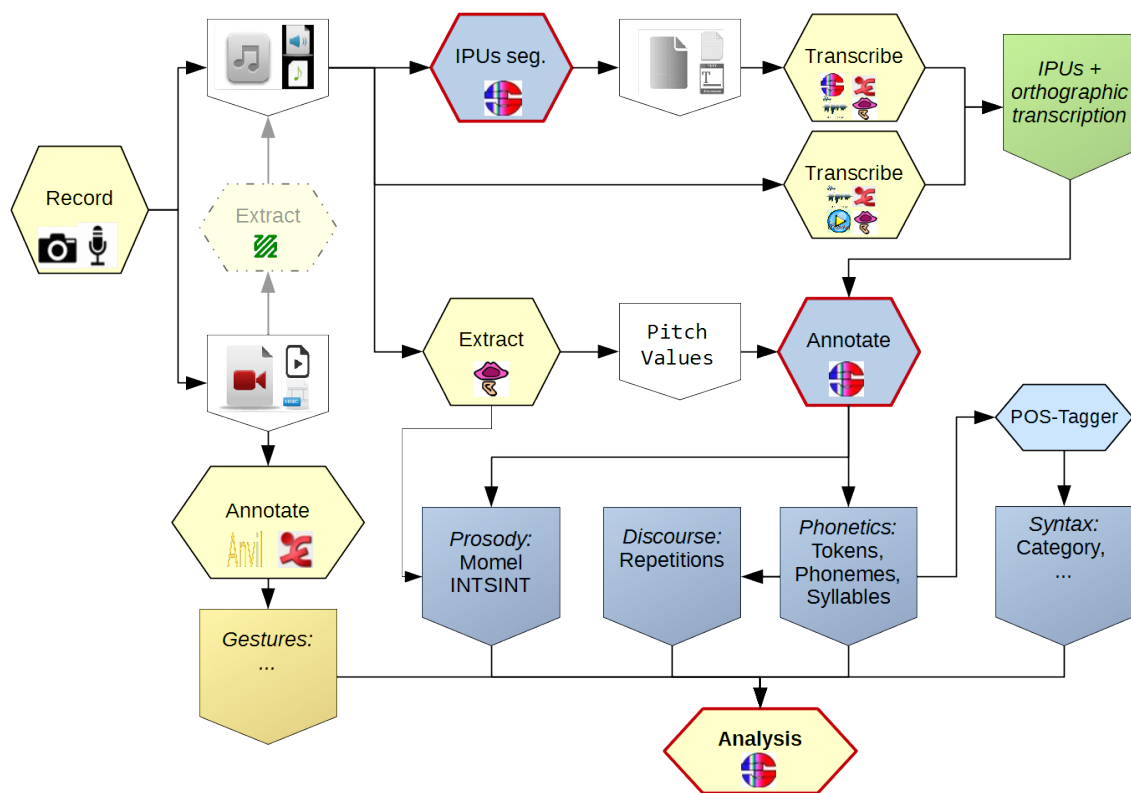


Figure 3.1: Annotation methodology

(IPUs) which are sounding segments surrounded by silent pauses of more than X ms, and time-aligned on the speech signal.

An orthographic transcription has to be performed **manually inside the IPUs** (do not expect to use an automatic speech transcription system). Then text normalization automatic annotation will normalize the orthographic transcription. The phonetization process will convert the normalized text in a grammar of pronunciations using X-SAMPA standard. Alignment will perform segmentation at phonemes and tokens levels, etc.

At the end of each automatic annotation process, SPPAS produces a *Procedure Outcome Report*. It contains important information about the annotations. It includes all parameters and eventually warnings and errors that occurred during the annotation process. This window opens in the scope to be read by users (!) and should be saved with the annotated corpus.

3.1.3 Recordings

SPPAS performs automatic annotations: It does not make sense to hope for miracles but you can expect good enough results that will allow you to save your precious time! And *it begins by taking care of the recordings...*

Audio files

Only `wav` and `au` audio file formats are supported by Python, so does SPPAS;

Only mono audio files are supported by automatic annotations of SPPAS.

SPPAS verifies if the audio file is 16 bits sample rate and 16000 Hz frame rate. Otherwise it automatically creates a new converted audio file. For very long files, this process may take time. If Python can't read the audio file, an error message is displayed: you'll have to convert it with audacity, praat... A relatively good recording quality is expected (see next Figure).

For example, both "Search for IPUs" and "Fill in IPUs" require a better quality compared to what is expected by "Alignment", and for that latter, it depends on the language. The quality of the result of automatic annotations highly depends on the quality of the audio file.

Providing a guideline or recommendation of good practices is impossible, because it depends on too many factors. However, the followings are obvious:

- Never, never, never records a lossy audio file. It means that extracting the audio file from a video is feasible only if the embedded audio is either lossless or not compressed: see this page https://en.wikipedia.org/wiki/Comparison_of_video_container_formats.
- The better microphone, the better audio file! Using an headworn microphone is much more better than a clip-on one. At LPL, we get very good results with the AKG C520.
- The recorded volume must be high enough. Ideally, it should be in range [-0.5 ; 0.5]. If all the amplitude values are in range [-0.1 ; 0.1], the difference between speech and silence is very slight and it makes the search for silences very difficult.
- The audio file should not be 32 floating bits. For speech, 32 bits are totally unusefull and - worse, sometimes Python can't read it.
- as you probably don't plan to burn your audio file on a CD-ROM, 44100Hz framerate does not make sense. 48000hz is a more reasonable choice, particularly because it doesn't need of elabotated interpolation methods when it's converted to 16000hz for automatic annotations.

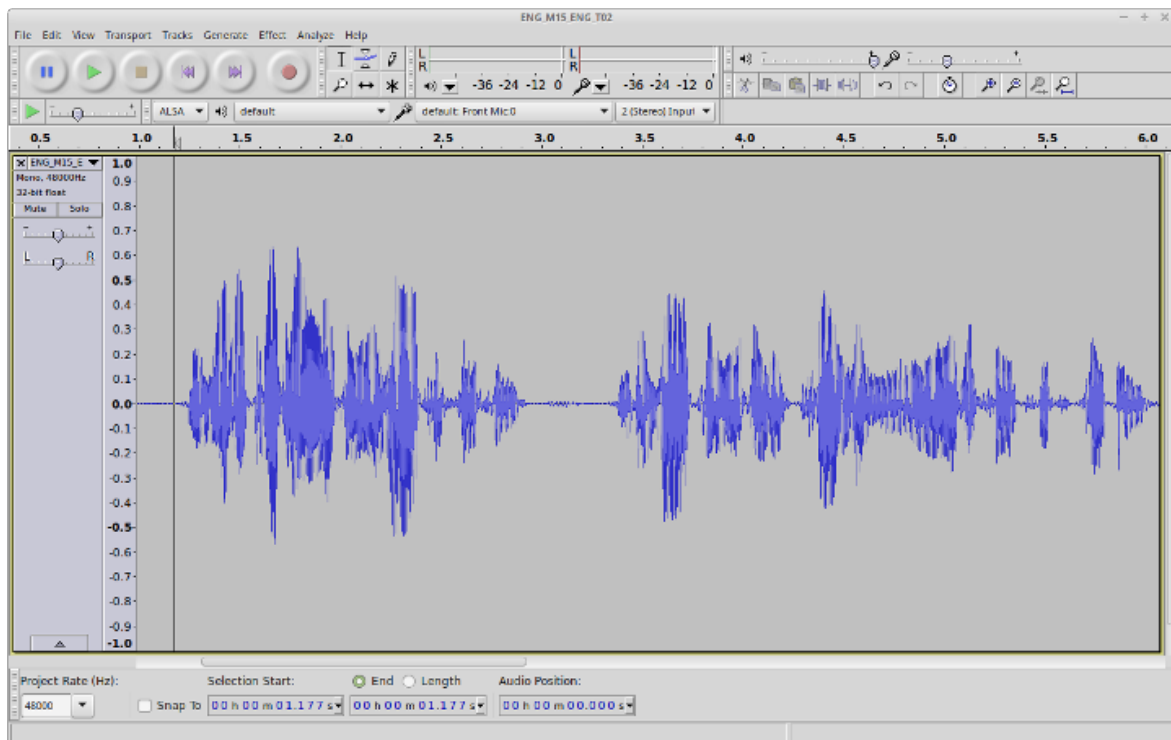


Figure 3.2: Example of expected recorded speech

Video files

SPPAS is proposing a few automatic annotations of a video if the Python library “opencv” is installed. All of them are annotating the face of recorded people.

3.1.4 File formats and tier names

When annotating with the GUI, the file names of each annotation is fixed and can’t be customized. A filename is made of a root, followed by a pattern then an extension. For example “oriana1-align.TextGrid” is made of the root “oriana1”, the pattern “-align” and the extension “.TextGrid”. Each annotation allows to fix manually the pattern and to choose the extension among the list of the supported ones. Notice that the pattern must start with the “-” (minus) character. It means that the character “-” must only be used to separate the root to the pattern:

The character ‘-’ can’t be used in the root of a filename.

The name of the tiers the annotations are expecting for their input are fixed and can’t be changed; so does the produced tier names.

3.2 Resources required to annotate

All the automatic annotations proposed by SPPAS are designed with language-independent algorithms, but some annotations are requiring language-knowledges. This linguistic knowledge is represented in external files so they can be added, edited or removed easily.

Adding a new language for a given annotation only consists in adding the linguistic resources the annotation needs, like lexicons, dictionaries, models, set of rules, etc. For exemple, see:

Mélanie Lancien, Marie-Hélène Côté, Brigitte Bigi (2020). Developing Resources for Automated Speech Processing of Quebec French. In Proceedings of The 12th Language Resources and Evaluation Conference, pp. 5323–5328, Marseille, France.

Brigitte Bigi, Bernard Caron, Abiola S. Oyelere (2017). Developing Resources for Automated Speech Processing of the African Language Naija (Nigerian Pidgin). In 8th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 441–445, Poznań, Poland.

3.2.1 Download and install linguistic resources

Since June 2020, the linguistic resources and models for some annotations are no longer distributed into the package of SPPAS. Instead, they are hosted by Ortolang repository with a public access.

They can be installed automatically into SPPAS by the `preinstall.py` program (CLI) or in the GUI by clicking “Add languages” or “Add annotations” in the toolbar of the “Annotate” page.

They can also be installed manually by downloading them at: <https://hdl.handle.net/11403/sppasresources> and unpacking the zip file into the `resources` folder of SPPAS package.

A full description of such resources and how to install them is available in the repository: download and read the file `Documentation.pdf`. It contains details about the list of phonemes, authors, licenses, etc.

3.2.2 New language support

Some of the annotations are requiring external linguistic resources in order to work efficiently on a given language: text normalization requires a lexicon, phonetization requires a pronunciation dictionary, etc. It is either possible to install and use the existing resources or to create and use custom ones.

When executing SPPAS, the list of available languages of each annotation is dynamically created by exploring the `resources` directory content. This means that:

- the resources you added or modified are automatically taken into account (ie. there’s no need to modify the program itself);
- SPPAS needs to be re-started if new resources are added when it was already being running.

3.3 Annotate with the GUI

Performing automatic annotations with SPPAS Graphical User Interface is a step-by-step process.

It starts by checking the list of paths and/or roots and/or files in the currently active workspace of the “Files” page. Then, in the “Annotate” page:

1. Select the output file format, i.e. the file format of the files SPPAS will create;
2. Select a language in the list;

3. Enable each annotation to perform by clicking on the button in red, among STANDALONE, SPEAKER and INTERACTION annotation types. Each button will be turned green if some annotations are selected.
 - 3.1 Configure each annotation by clicking on the “Configure...” link text in blue;
 - 3.2 The language of any annotation can be changed.
4. Click on the *Perform annotations* button, and wait. A progress bar should indicates the annotation steps and files. Some annotations are very very fast but some others are not. For example, Face Detection is 2.5 x real times, i.e. annotating a video of 1 minute will take 2 minutes 30 secs.
5. It is important to read the Procedure Outcome report. It allows to check that everything happened normally during the automatic annotations. This report is saved in the “logs” folder of the SPPAS package.

3.4 Annotate with the CLI

To perform automatic annotations with the Command-line User Interface, there is a main program `annotation.py`. This program allows to annotate in an easy-and-fast way but none of the annotations can be configured: their default parameters are used. This program performs automatic annotations on a given file or on all files of a directory. It strictly corresponds to the button *Perform annotations* of the GUI except that annotations are pre-configured: no specific option can be specified.

```
usage: python .\sppas\bin\annotation.py -I file|folder [options]
```

optional arguments:

```
-h, --help            show this help message and exit
--log file            File name for a Procedure Outcome Report (default: None)
--momel              Activate Momel
--intsint            Activate INTSINT
--fillipus           Activate Fill in IPUs
--searchipus         Activate Search for IPUs
--textnorm           Activate Text Normalization
--phonetize          Activate Phonetization
--alignment          Activate Alignment
--syllabify           Activate Syllabification
--tga                Activate Time Group Analysis
--activity           Activate Activity
--rms                Activate RMS
--selfrepet          Activate Self-Repetitions
--stopwords           Activate Stop Tags
--lexmetric          Activate LexMetric
--otherrepet         Activate Other-Repetitions
--reoccurrences      Activate Re-Occurrences
--merge              Create a merged file with all the annotations
```

Files:

```
-I file|folder        Input transcription file name (append).
-l lang               Language code (iso8859-3). One of: por eng ita kor deu nan
                     vie und hun spa cat pol yue fra pcm yue_chars cmn jpn.
-e .ext               Output file extension. One of: .xra .TextGrid .eaf .csv
                     .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff
```

```

Brigitte@asus-bigi:/cygdrive/d/workspace/SPPAS$ ./bin/annotation.py -i ./samples/samples-eng -l eng --ipu --tok --phon --align
SPPAS - Version 1.6.2
Copyright (C) 2011-2014 LPL Laboratory
http://www.lpl-aix.fr/~bigi/sppas/

=====
100% [=====] IPUs Segmentation
Finished.

100% [=====] Tokenization
Finished.

100% [=====] Phonetization
Finished.

100% [=====] Alignment
Finished.

SPPAS finished.
See ./samples/samples-eng.log for details.
Thank you for using SPPAS.
Brigitte@asus-bigi:/cygdrive/d/workspace/SPPAS$ ;

```

Figure 3.3: CLI: annotation.py output example

Examples of use:

```

./sppas/bin/annotation.py -I .\samples\samples-eng
                        -l eng
                        -e .TextGrid
                        --fillipus --textnorm --phonetize --alignment

```

A progress bar is displayed for each annotation if the Terminal is supporting it (bash for example). Instead, the progress is indicated line-by-line (Windows PowerShell for example).

Each annotation has also its own program and all options can be fixed. They are all located in the `sppas/bin` folder.

3.5 The procedure outcome report

It is very important to read conscientiously this report: it describes exactly what happened during the automatic annotation process. It is recommended to store a copy of the report within the corpus because it contains information that are interesting to know for anyone using the annotations.

By default, all reports are saved in the “logs” folder of the SPPAS package.

The text first indicates the version of SPPAS that was used. This information is very important. Annotations in SPPAS and their related resources are regularly improved and then, the result of the automatic process can change from one version to the other one.

Example:

```

SPPAS version 3.5
Copyright (C) 2011-2021 Brigitte Bigi
Web site: http://www.sppas.org/
Contact: Brigitte Bigi (contact@sppas.org)

```

Secondly, the text shows information related to the given input:

1. the selected language of each annotation - only if the annotation is language-dependent. For some language-dependent annotations, SPPAS can still perform the annotation even if the resources for a given language are not available: in that case, select “und”, which is the iso639-3 code for “undetermined”.
2. the selected files and folder to be annotated.
3. the list of annotations, and if each annotation was enabled. In that case, enabled means that the checkbox of the annotation was checked by the user and that the resources are available for the given language. On the contrary, disabled means that either the checkbox was not checked or the required resources are not available.
4. the file format of the resulting files.

Example:

Date: 2020-04-21T11:14:01+02:00

Input languages:

- Momel: ---
- INTSINT: ---
- Fill in IPU: ---
- Search for IPU: ---
- Text Normalization: eng
- Phonetization: eng
- Alignment: eng
- Syllabification:
- Time Group Analysis: ---
- Activity: ---
- RMS: ---
- Self-Repetitions:
- Stop Tags:
- LexMetric: ---
- Other-Repetitions:
- Re-Occurrences: ---

Selected files and folders:

- oriana1.wav

Selected annotations:

- Momel: enabled
- INTSINT: enabled
- Fill in IPU: enabled
- Search for IPU: disabled
- Text Normalization: enabled
- Phonetization: enabled
- Alignment: enabled
- Syllabification: disabled
- Time Group Analysis: disabled
- Activity: disabled
- RMS: disabled
- Self-Repetitions: disabled
- Stop Tags: disabled
- LexMetric: disabled
- Other-Repetitions: disabled

- Re-Occurrences: disabled

File extension: .xra

Thirdly, each automatic annotation is described in details, for each annotated file. At a first stage, the list of options and their value is summarized. Example:

Text Normalization

```
The vocabulary contains 121250 tokens.
The replacement dictionary contains 8 items.
Options:
... inputpattern:
... outputpattern: -token
... faked: True
... std: False
... custom: False
... occ_dur: True
```

Then, a diagnosis of the given file is printed. It can be: 1. “Valid”: the file is relevant 2. “Admit”: the file is not like expected but SPPAS will convert it and work on the converted file. 3. “Invalid”: SPPAS can’t work with that file. The annotation is then disabled. In case 2 and 3, a message indicates the origin of the problem.

Then, if any, the annotation procedure prints messages. Four levels of information must draw your attention:

1. “[OK]” means that everything happened normally. The annotation was performed successfully.
2. “[IGNORE]” means that SPPAS ignored the file and didn’t do anything.
3. “[WARNING]” means that something happened abnormally, but SPPAS found a solution, and the annotation was performed anyway.
4. “[ERROR]” means that something happened abnormally and SPPAS failed to found a solution. The annotation was either not performed, or performed with a wrong result.

Example of “Warning” message:

```
... ... Export AP_track_0711.TextGrid
... ... into AP_track_0711.xra
... ... [ IGNORE ] because a previous segmentation is existing.
```

Example of “Warning” message:

```
... ... [ WARNING ] chort- is missing of the dictionary and was
... ... automatically phonetized as S-O/-R-t
```

At the end of the report, the “Result statistics” section mentions the number of files that were annotated for each annotation, or -1 if the annotation was disabled.

3.6 Orthographic Transcription

An orthographic transcription is often the minimum requirement for a speech corpus so it is at the top of the annotation procedure, and it is the entry point for most of the automatic annotations. A *transcription convention* is designed to provide rules for writing speech corpora. This convention establishes what are the phenomena to transcribe and also how to mention them in the orthography.

From the beginning of its development it was considered to be essential for SPPAS to deal with an **Enriched Orthographic Transcription** (EOT). The transcription convention is summarized below and all details are given in the file `TOE-SPPAS.pdf`, available in the `documentation` folder. It indicates the rules and includes examples of what is expected or recommended.

Convention overview:

- truncated words, noted as a ‘-’ at the end of the token string (an ex-ample);
- noises, noted by a ‘*’ (not available for some languages);
- laughter, noted by a ‘@’ (not available for some languages);
- short pauses, noted by a ‘+’;
- elisions, mentioned in parenthesis;
- specific pronunciations, noted with brackets [example,eczap];
- comments are preferably noted inside braces {this is a comment!};
- comments can be noted inside brackets without using comma [this and this];
- liaisons, noted between ‘=’ (this =n= example);
- morphological variants with <ice scream,I scream>;
- proper name annotation, like \$ John S. Doe \$.

The symbols * + @ must be surrounded by whitespace.

SPPAS allows to include the regular punctuations. For some languages, it also allows to include numbers: they will be automatically converted to their written form during Text Normalization process.

From this EOT, several derived transcriptions can be generated automatically, including the two followings:

1. the standard transcription is the list of orthographic tokens (optional);
2. a specific transcription from which the phonetic tokens are obtained to be used by the grapheme-phoneme converter that is named faked transcription (the default).

For example, with the transcribed sentence: *This [is,iz] + hum... an enrich(ed) transcription {loud} number 1!*, the derived transcriptions are:

1. standard: *this is + hum an enriched transcription number one*
2. tokens: *this iz + hum an enrich transcription number one*

Notice that the convention allows to include a large scale of phenomena, for which most of them are optional. As a minimum, the **transcription must include**:

- filled pauses;
- short pauses;
- repeats;

- noises and laugh items (not available for Japanese and Cantonese).

Finally, it has to be noticed that this convention is not software-dependent. The orthographic transcription can be manually performed within SPPAS GUI in the “Edit” page, with Praat, with Annotation Pro, Audacity, ...

3.7 Search for Inter-Pausal Units (IPUs)

3.7.1 Overview

The Search for IPUs is a semi-automatic annotation process. This segmentation provides an annotated file with one tier named “IPUs”. The silence intervals are labelled with the “#” symbol, and IPUs intervals are labelled with “ipu_” followed by the IPU number. This annotation is semi-automatic: **it should be verified manually**.

Notice that the better recording quality, the better IPUs segmentation.

3.7.2 The parameters

The following parameters must be properly fixed:

- Minimum volume value (in seconds): If this value is set to zero, the minimum volume is automatically adjusted for each sound file. Try with it first, then if the automatic value is not correct, set it manually. The Procedure Outcome Report indicates the value the system choose. The AudioRoamer component can also be of great help: it indicates min, max and mean volume values of the sound.
- Minimum silence duration (in seconds): By default, this is fixed to 0.2 sec. This duration mostly depends on the language. It is commonly fixed to at least 0.2 sec for French and at least 0.25 seconds for English language.
- Minimum speech duration (in seconds): By default, this value is fixed to 0.3 sec. A relevant value depends on the speech style: for isolated sentences, probably 0.5 sec should be better, but it should be about 0.1 sec for spontaneous speech.
- IPUs boundary shift (in seconds) for start or end: a duration which is systematically added to IPUs boundaries, to enlarge the IPUs interval, and as a consequence, the neighboring silences are reduced.

The procedure outcome report indicates the values (volume, minimum durations) that were used by the system for each sound file.

3.7.3 Perform “Search for IPUs” with the GUI

It is an annotation of STANDALONE type.

Click on the “Search IPUs” activation button and on the “Configure...” blue text to fix options.

Notice that the speech segments can be transcribed using SPPAS, in the “Analyze” page.

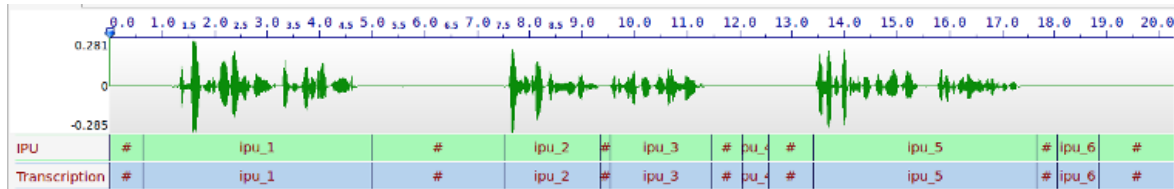


Figure 3.4: Example of result

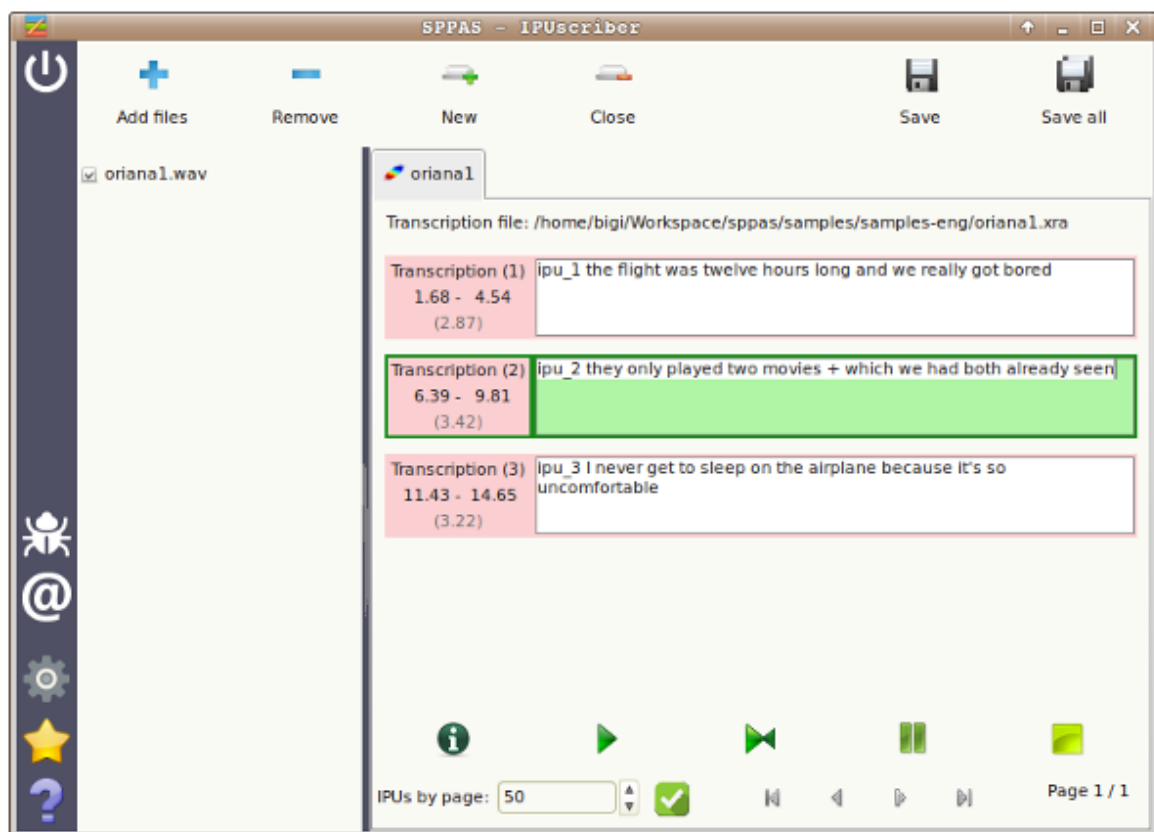


Figure 3.5: Orthographic transcription based on IPUs

3.7.4 Perform “Search for IPUs” with the CLI

`searchipus.py` is the program to perform this semi-automatic annotation, i.e. silence/IPUs segmentation, either on a single file (-i and optionnally -o) or on a set of files (by using -I and optionnally -e).

Usage

```
searchipus.py [files] [options]
```

Search for IPUs: Search for Inter-Pausal Units in an audio file.

optional arguments:

-h, --help	show this help message and exit
--quiet	Disable the verbosity
--log file	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

-i file	Input wav file name.
-o file	Annotated file with silences/units segmentation (default: None)

Files (auto mode):

-I file	Input wav file name (append).
-e .ext	Output file extension. One of: .xra .TextGrid .eaf .csv .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff

Options:

--outputpattern OUTPUTPATTERN	Output file pattern (default:)
--win_length WIN_LENGTH	Window size to estimate rms (in seconds) (default: 0.020)
--threshold THRESHOLD	Threshold of the volume value (rms) for the detection of silences, 0=automatic (default: 0)
--min_ipu MIN_IPU	Minimum duration of an IPU (in seconds) (default: 0.300)
--min_sil MIN_SIL	Minimum duration of a silence (in seconds) (default: 0.200)
--shift_start SHIFT_START	Systematically move at left the boundary of the beginning of an IPU (in seconds) (default: 0.01)
--shift_end SHIFT_END	Systematically move at right the boundary of the end of an IPU (in seconds) (default: 0.02)

This program is part of SPPAS version 2.4. Copyright (C) 2011–2019 Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

A single input file and output on stdout:


```
python .\sppas\bin\searchipus.py -i .\samples\samples-eng\oriana1.wav
2018-12-19 10:49:32,782 [INFO] Logging set up level=15
2018-12-19 10:49:32,790 [INFO] ... Information:
2018-12-19 10:49:32,792 [INFO] ... ... Number of IPUs found:          3
2018-12-19 10:49:32,792 [INFO] ... ... Threshold volume value:      0
2018-12-19 10:49:32,792 [INFO] ... ... Threshold silence duration: 0.200
2018-12-19 10:49:32,792 [INFO] ... ... Threshold speech duration:  0.300
0.000000 1.675000 #
1.675000 4.580000 ipu_1
4.580000 6.390000 #
6.390000 9.880000 ipu_2
9.880000 11.430000 #
11.430000 14.740000 ipu_3
14.740000 17.792000 #
```

Idem without logs:

```
python .\sppas\bin\searchipus.py -i .\samples\samples-eng\oriana1.wav --quiet
0.000000 1.675000 #
1.675000 4.580000 ipu_1
4.580000 6.390000 #
6.390000 9.880000 ipu_2
9.880000 11.430000 #
11.430000 14.740000 ipu_3
14.740000 17.792000 #
```

Several input files, output in Praat-TextGrid file format:

```
python .\sppas\bin\searchipus.py -I .\samples\samples-eng\oriana1.wav \
-I .\samples\samples-eng\oriana3.wave -e .TextGrid
2018-12-19 10:48:16,520 [INFO] Logging set up level=15
2018-12-19 10:48:16,522 [INFO] File oriana1.wav: Valid.
2018-12-19 10:48:16,532 [INFO] ... Information:
2018-12-19 10:48:16,532 [INFO] ... ... Number of IPUs found:          3
2018-12-19 10:48:16,532 [INFO] ... ... Threshold volume value:      0
2018-12-19 10:48:16,532 [INFO] ... ... Threshold silence duration: 0.200
2018-12-19 10:48:16,533 [INFO] ... ... Threshold speech duration:  0.300
2018-12-19 10:48:16,538 [INFO] ... E:\bigi\Projets\sppas\samples\samples-eng\oriana1.TextGrid
2018-12-19 10:48:16,538 [INFO] File oriana3.wave: Invalid.
2018-12-19 10:48:16,539 [ERROR] ... ... An audio file with only one channel is expected. Got
2018-12-19 10:48:16,540 [INFO] ... No file was created.
```

3.8 Fill in Inter-Pausal Units (IPUs)

3.8.1 Overview

This automatic annotation consists in aligning macro-units of a document with the corresponding sound.

IPUs are blocks of speech bounded by silent pauses of more than X ms. This annotation searches for a silences/IPUs segmentation of a recorded file (see previous section) and fill in the IPUs with the transcription given in a `txt` file.

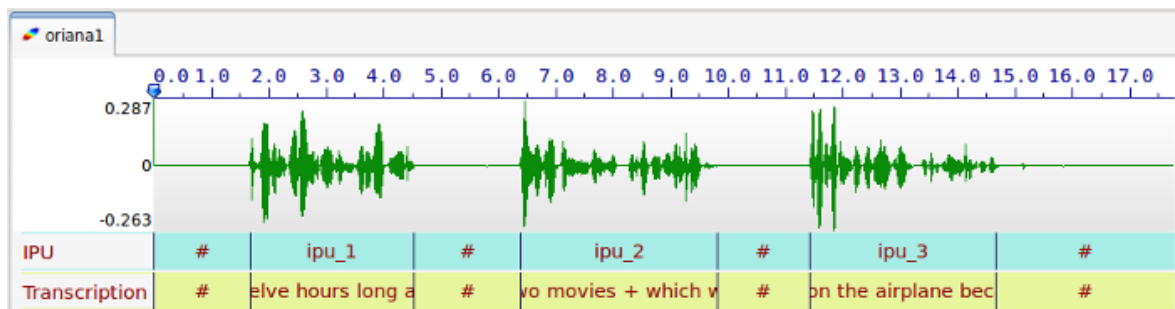


Figure 3.6: Fill in IPUs

3.8.2 How does it work

SPPAS identifies silent pauses in the signal and attempts to align them with the units proposed in the transcription file, under the assumption that each such unit is separated by a silent pause. It is based on the search of silences described in the previous section, but in this case, the number of units to find is known. The system adjusts automatically the volume threshold and the minimum durations of silences/IPUs to get the right number of units. The content of the units has no regard, because SPPAS does not interpret them: it can be the orthographic transcription, a translation, numbers, ... This algorithm is language-independent: it can work on any language.

In the transcription file, **silent pauses must be indicated** using both solutions, which can be combined:

- with the symbol '#';
- with newlines.

A recorded speech file must strictly correspond to a `txt` file of the transcription. The annotation provides an annotated file with one tier named "Transcription". The silence intervals are labelled with the "#" symbol, as IPUs are labelled with "ipu_" followed by the IPU number then the corresponding transcription.

The same parameters than those indicated in the previous section must be fixed.

Remark: This annotation was tested on read speech no longer than a few sentences (about 1 minute speech) and on recordings of very good quality.

3.8.3 Perform "Fill in IPUs" with the GUI

It is an annotation of STANDALONE type.

Click on the "Fill in IPUs" activation button and on the "Configure..." blue text to fix options.

3.8.4 Perform "Fill in IPUs" with the CLI

`fillipus.py` is the program to perform this IPUs segmentation, i.e. silence/ipus segmentation, either on a single file (-i and optionnally -o) or on a set of files (by using -I and optionnally -e).

Usage

```
fillipus.py [files] [options]
```

Fill in IPUs: Search for Inter-Pausal Units and fill in with a transcription.
Requires an audio file and a .txt file with the transcription.

optional arguments:

```
-h, --help            show this help message and exit
--quiet               Disable the verbosity
--log file            File name for a Procedure Outcome Report (default: None)
```

Files (manual mode):

```
-i file              Input wav file name.
-t file              Input transcription file name.
-o file              Annotated file with filled IPUs
```

Files (auto mode):

```
-I file              Input wav file name (append).
-e .ext              Output file extension. One of: .xra .TextGrid .eaf .csv
                    .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff
```

Options:

```
--outputpattern OUTPUTPATTERN
                    Output file pattern (default: )
--min_ipu MIN_IPU   Initial minimum duration of an IPU (in seconds) (default:
                    0.300)
--min_sil MIN_SIL   Initial minimum duration of a silence (in seconds)
                    (default: 0.200)
```

This program is part of SPPAS version 3.0. Copyright (C) 2011-2020 Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

A single input file with an input in manual mode:

```
python .\sppas\bin\fillipus.py -i .\samples\samples-eng\oriana1.wav -t .\samples\samples-eng\oriana1.txt
2018-12-19 11:03:15,614 [INFO] Logging set up level=15
2018-12-19 11:03:15,628 [INFO] ... Information:
2018-12-19 11:03:15,628 [INFO] ... .. Threshold volume value:      122
2018-12-19 11:03:15,630 [INFO] ... .. Threshold silence duration: 0.200
2018-12-19 11:03:15,630 [INFO] ... .. Threshold speech duration:  0.300
0.000000 1.675000 #
1.675000 4.570000 the flight was 12 hours long and we really got bored
4.570000 6.390000 #
6.390000 9.870000 they only played two movies + which we had both already seen
9.870000 11.430000 #
11.430000 14.730000 I never get to sleep on the airplane because it's so uncomfortable
14.730000 17.792000 #
```

A single input file in automatic mode:

```
python .\sppas\bin\fillipus.py -I .\samples\samples-eng\oriana1.wav
```

```
python .\sppas\bin\fillipus.py -I .\samples\samples-eng\oriana1.wav
python .\sppas\bin\fillipus.py -I .\samples\samples-eng\oriana1.txt
```

3.9 Text normalization

3.9.1 Overview

In principle, any system that deals with unrestricted text need the text to be normalized. Texts contain a variety of “non-standard” token types such as digit sequences, words, acronyms and letter sequences in all capitals, mixed case words, abbreviations, roman numerals, URL’s and e-mail addresses... Normalizing or rewriting such texts using ordinary words is then an important issue. The main steps of the text normalization implemented in SPPAS (Bigi 2011) are:

- Replace symbols by their written form, thanks to a “replacement” dictionary, located into the folder “repl” in the “resources” directory.
- Word segmentation based on the content of a lexicon.
- Convert numbers to their written form.
- Remove punctuation.
- Lower the text.

3.9.2 Adapt Text normalization

Word segmentation of SPPAS is mainly based on the use of a lexicon. If a segmentation is not as expected, it is up to the user to modify the lexicon: Lexicons of all supported languages are all located in the folder “vocab” of the “resources” directory. They are in the form of “one word at a line” with [UTF-8 encoding](#) and [“LF” for newline](#).

3.9.3 Support of a new language

Adding a new language in Text Normalization consists in the following steps:

1. Create a lexicon. Fix properly its encoding (utf-8), its newlines (LF), and fix the name and extension of the file as follow:
 - language name with iso639-3 standard
 - extension “.vocab”
2. Put this lexicon in the `resources/vocab` folder
3. Create a replacement dictionary for that language (take a look on the ones of the other language!)
4. Optionally, the language can be added into the `num2letter.py` program

That’s it for most of the languages! If the language requires more steps, simply write to the author to collaborate, find some funding, etc. like it was already done for Cantonese (Bigi & Fung 2015) for example.

3.9.4 Perform Text Normalization with the GUI

It is an annotation of STANDALONE type.

The SPPAS Text normalization system takes as input a file (or a list of files) for which the name strictly match the name of the audio file except the extension. For example, if a file with name “oriana1.wav” is given, SPPAS will search for a file with name “oriana1.xra” at a first stage if “.xra” is set as the default extension, then it will search for other supported extensions until a file is found.

This file must include a tier with an orthographic transcription. At a first stage, SPPAS tries to find a tier with `transcription` as name. If such a tier does not exist, the first tier that is matching one of the following strings is used (case-insensitive search):

1. `trans`
2. `trs`
3. `ipu`
4. `ortho`
5. `toe`

Text normalization produces a file with “-token” appended to its name, i.e. “oriana1-token.xra” for the previous example. By default, this file is including only one tier with the resulting normalization and with name “Tokens”. To get other versions of the normalized transcription, click on the “Configure” text then check the expected tiers.

Read the “Introduction” of this chapter for a better understanding of the difference between “standard” and “faked” results.

To perform the text normalization process, click on the Text Normalization activation button, select the language and click on the “Configure...” blue text to fix options.

3.9.5 Perform Text Normalization with the CLI

`normalize.py` is the program to perform Text Normalization, i.e. the text normalization of a given file or a raw text.

Usage

```
normalize.py [files] [options]
```

Text Normalization: Text normalization segments the orthographic transcription into tokens and remove punctuation, convert numbers, etc. Requires an orthographic transcription into IPUs.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

<code>-i file</code>	Input transcription file name.
<code>-o file</code>	Annotated file with normalized tokens.

Files (auto mode):

```
-I file          Input transcription file name (append).
-l lang          Language code (iso8859-3). One of: cat cmn deu eng fra hun
                  ita jpn kor nan pcm pol por spa vie yue yue_chars.
-e .ext          Output file extension. One of: .xra .TextGrid .eaf .csv
                  .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff
```

Resources:

```
-r vocab          Vocabulary file name
```

Options:

```
--inputpattern INPUTPATTERN
                        Input file pattern (orthographic transcription)
                        (default: )
--outputpattern OUTPUTPATTERN
                        Output file pattern (default: -token)
--faked FAKED          Create a tier with the faked tokens (required for
                        phonetization) (default: True)
--std STD              Create a tier with the standard tokens (useful if EOT)
                        (default: False)
--custom CUSTOM        Create a customized tier (default: False)
--occ_dur OCC_DUR      Create tiers with number of tokens and duration of
                        each IPU (default: True)
```

This program is part of SPPAS version 2.4. Copyright (C) 2011–2019 Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

A single input file with a raw transcription input in manual mode:

```
python .\sppas\bin\normalize.py -r .\resources\vocab\eng.vocab -i .\samples\samples-eng\oriana1
2018-12-19 11:48:34,151 [INFO] Logging set up level=15
2018-12-19 11:48:34,473 [INFO] ... .. Intervalle numéro 1.
2018-12-19 11:48:34,477 [INFO] ... .. Intervalle numéro 2.
2018-12-19 11:48:34,480 [INFO] ... .. Intervalle numéro 3.
Tokens
1, the flight was twelve hours long and we really got bored
2, they only played two movies + which we had both already seen
3, i never get to sleep on the airplane because it's so uncomfortable
```

A single input file with a transcription time-aligned into the IPUS, in manual mode and no logs:

```
python .\sppas\bin\normalize.py -r .\resources\vocab\eng.vocab
-i .\samples\samples-eng\oriana1.xra --quiet
Tokens
0.000000, 1.675000 #
1.675000, 4.570000 the flight was twelve hours long and we really got bored
4.570000, 6.390000 #
6.390000, 9.870000 they only played two movies + which we had both already seen
9.870000, 11.430000 #
11.430000, 14.730000 i never get to sleep on the airplane because it's so uncomfortable
14.730000, 17.792000 #
```

The same file in automatic mode can be annotated with one of the following commands:

```
python .\sppas\bin\normalize.py -I .\samples\samples-eng\oriana1.xra -l eng
python .\sppas\bin\normalize.py -I .\samples\samples-eng\oriana1.txt -l eng
python .\sppas\bin\normalize.py -I .\samples\samples-eng\oriana1.wav -l eng
python .\sppas\bin\normalize.py -I .\samples\samples-eng\oriana1 -l eng
```

This program can also normalize data from the standard input. Example of use, using stdin/stdout under Windows:

```
Write-Output "The flight was 12 HOURS {toto} long." |
python .\sppas\bin\normalize.py -r .\resources\vocab\eng.vocab --quiet
the
flight
was
twelve
hours
long
```

In that case, the comment mentioned with the braces is removed and the number is converted to its written form. The character “_” is used for compound words (it replaces the whitespace).

3.10 Phonetization

3.10.1 Overview

Phonetization, also called grapheme-phoneme conversion, is the process of representing sounds with phonetic signs. However, converting from written text into actual sounds, for any language, cause several problems that have their origins in the relative lack of correspondence between the spelling of the lexical items and their sound contents. As a consequence, SPPAS implements a dictionary based-solution which consists in storing a maximum of phonological knowledge in a lexicon. This approach is then language-independent. SPPAS phonetization process is the equivalent of a sequence of dictionary look-ups.

Most of the other systems assume that all words of the speech transcription are mentioned in the pronunciation dictionary. On the contrary, SPPAS includes a language-independent algorithm which is able to phonetize unknown words of any language as long as a (minimum) dictionary is available (Bigi 2013). The Procedure Outcome Report reports on such cases with a WARNING message.

3.10.2 Adapt Phonetization

Since Phonetization is only based on the use of a pronunciation dictionary, the quality of the result only depends on this resource. If a pronunciation is not as expected, it is up to the user to change it in the dictionary: Dictionaries are located in the folder “dict” of the “resources” directory. They are all with [UTF-8 encoding](#) and [“LF” for newline](#). The format of the dictionaries is HTK-like. As example, below is a piece of the eng.dict file:

THE	[THE]	D @
THE (2)	[THE]	D V
THE (3)	[THE]	D i:

THEA	[THEA]	T i: @
THEALL	[THEALL]	T i: l
THEANO	[THEANO]	T i: n @U
THEATER	[THEATER]	T i: @ 4 3:r
THEATER'S	[THEATER'S]	T i: @ 4 3:r z

The first column indicates the word, followed by the variant number (except for the first one). The second column indicates the word between brackets. The last columns are the succession of phones, separated by a whitespace. SPPAS is relatively compliant with the format and accept empty brackets or missing brackets.

The phoneset of the languages are mainly based on [X-SAMPA](#) international standard. See the chapter “Resources” of this documentation to know the list of accepted phones for a given language. This list can’t be extended nor modified by users. However, new phones can be added: Send an e-mail to the author to collaborate in that way.

Actually, some words can correspond to several entries in the dictionary with various pronunciations. These pronunciation variants are stored in the phonetization result. By convention, whitespace separate words, minus characters separate phones and pipe character separate phonetic variants of a word. For example, the transcription utterance:

- **Transcription:** The flight was 12 hours long.
- **Text Normalization:** the flight was twelve hours long
- **Phonetization:** D-@|D-V|D-i: f-l-aI-t w-A-z|w-V-z|w-@-z|w-O:-z t-w-E-l-v
aU-3:r-z|aU-r-z l-O:-N

3.10.3 Support of a new language

The support of a new language in Phonetization only consists in: 1. creating the pronunciation dictionary. The following constraints on the file must be respected: - its format (HTK-like), - its encoding (UTF-8), - its newlines (LF), - its phone set (X-SAMPA), - its file name (iso639-3 of the language and “.dict” extension). 2. adding the dictionary in the “dict” folder of the “resources” directory.

3.10.4 Perform Phonetization with the GUI

It is an annotation of STANDALONE type.

The Phonetization process takes as input a file that strictly match the audio file name except for the extension and that “-token” is appended. For example, if the audio file name is “oriana1.wav”, the expected input file name is “oriana1-token.xra” if .xra is the default extension for annotations. This file must include a **normalized** orthographic transcription. The name of such tier must contains one of the following strings:

1. “tok”
2. “trans”

The first tier that matches one of these requirements is used (this match is case-insensitive).

Phonetization produces a file with “-phon” appended to its name, i.e. “oriana1-phon.xra” for the previous example. This file contains only one tier with the resulting phonetization and with name “Phones”.

To perform the annotation, click on the Phonetization activation button, select the language and click on the “Configure...” blue text to fix options.

3.10.5 Perform Phonetization with the CLI

`phonetize.py` is the program to perform Phonetization on a given file, i.e. the grapheme-conversion of a file or a raw text.

Usage

```
phonetize.py [files] [options]
```

Phonetization: Grapheme to phoneme conversion represents sounds with phonetic signs. Requires a Text Normalization.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

<code>-i file</code>	Input tokenization file name.
<code>-o file</code>	Annotated file with phonetization.

Files (auto mode):

<code>-I file</code>	Input transcription file name (append).
<code>-l lang</code>	Language code (iso8859-3). One of: cat cmn deu eng fra ita jpn kor nan pcm pol por spa yue yue_chars.
<code>-e .ext</code>	Output file extension. One of: .xra .TextGrid .eaf .csv .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff

Resources:

<code>-r dict</code>	Pronunciation dictionary (HTK-ASCII format).
<code>-m map_file</code>	Pronunciation mapping table. It is used to generate new pronunciations by mapping phonemes of the dictionary.

Options:

<code>--inputpattern INPUTPATTERN</code>	Input file pattern (tokenization) (default: -token)
<code>--outputpattern OUTPUTPATTERN</code>	Output file pattern (default: -phon)
<code>--unk UNK</code>	Try to phonetize unknown words (default: True)
<code>--usestdtokens USESTDtokens</code>	Phonetize from standard spelling (default: False)

This program is part of SPPAS version 2.4. Copyright (C) 2011-2019 Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

A single input file with a normalized text in manual mode:

```
python .\sppas\bin\phonetize.py -r .\resources\dict\eng.dict
-i .\samples\samples-eng\oriana1-token.xra --quiet
```

```

Phones
0.000000, 1.675000, sil
1.675000, 4.570000, {D-@|D-i:|D-V} f-l-aI-t {w-@-z|w-V-z|w-O:-z|w-A-z} t-w-E-l-v
{aU-3:r-z|aU-r\~z} l-O:-N {{-n-d|@-n-d} w-i: {r\~I-l-i:|r\~i:-l-i:} g-A-t b-O:-r\~d
4.570000, 6.390000, sil
6.390000, 9.870000, D-eI @U-n-l-i: p-l-eI-d t-u m-u-v-i:-z sil {h-w-I-tS|w-I-tS}
w-i: h-{-d b-@U-T {O:-l-r\~E-4-i:|O:-r\~E-4-i:} s-i:-n
9.870000, 11.430000, sil
11.430000, 14.730000, aI n-E-v-3:r {g-I-t|g-E-t} {t-@|t-i|t-u} s-l-i:-p
{O:-n|A-n} {D-@|D-i:|D-V} E-r\~p-l-eI-n {b-i-k-O:-z|b-i-k-V-z} {i-t-s|I-t-s}
s-@U @-n-k-V-m-f-3:r-4-@-b-@-l
14.730000, 17.792000, sil

```

The same file in automatic mode can be annotated with one of the following commands:

```

python .\sppas\bin\phonetize.py -l eng -I .\samples\samples-eng\oriana1-token.xra
python .\sppas\bin\phonetize.py -l eng -I .\samples\samples-eng\oriana1.xra
python .\sppas\bin\phonetize.py -l eng -I .\samples\samples-eng\oriana1.txt
python .\sppas\bin\phonetize.py -l eng -I .\samples\samples-eng\oriana1.wav
python .\sppas\bin\phonetize.py -l eng -I .\samples\samples-eng\oriana1

```

This program can also phonetize data from the standard input. Example of use, using stdin/stdout under Windows:

```

Write-Output "The flight was 12 HOURS {toto} long." |
python .\sppas\bin\normalize.py -r .\resources\vocab\eng.vocab --quiet |
python .\sppas\bin\phonetize.py -r .\resources\dict\eng.dict --quiet
D-@|D-V|D-i:
f-l-aI-t
w-A-z|w-V-z|w-@-z|w-O:-z
t-w-E-l-v
aU-3:r-z|aU-r\~z
l-O:-N

```

3.11 Alignment

3.11.1 Overview

Alignment, also called phonetic segmentation, is the process of aligning speech with its corresponding transcription at the phone level. The alignment problem consists in a time-matching between a given speech unit along with a phonetic representation of the unit.

SPPAS Alignment does not perform the segmentation itself. It is a wrapper either for the Julius Speech Recognition Engine (SRE) or for the `hVite` command of HTK-Toolkit. In addition, SPPAS can perform a “basic” alignment, assigning the same duration to each sound.

Speech Alignment requires an Acoustic Model in order to align speech. An acoustic model is a file that contains statistical representations of each of the distinct sounds of one language. Each sound is represented by one of these statistical representations. The quality of the alignment result only depends on both this resource and on the aligner. From our past experiences, we got better results with Julius. See the chapter 4 “Resources for Automatic Annotations” to get the list of sounds of each language.

Notice that SPPAS allows to time-align automatically laugh, noises, or filled pauses (depending on the language): No other system is able to achieves this task!

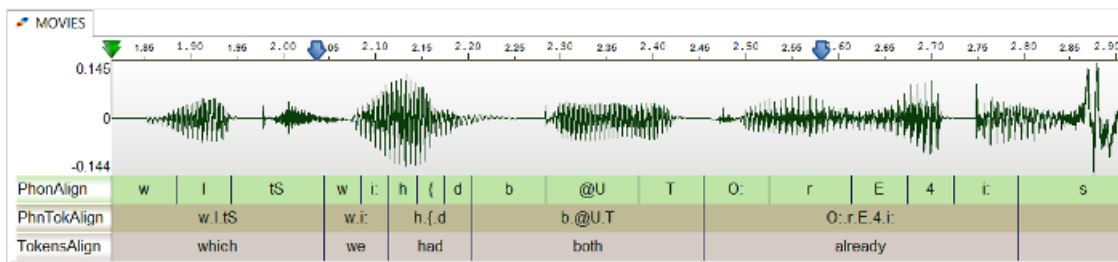


Figure 3.7: SPPAS alignment output example

3.11.2 Adapt Alignment

The better Acoustic Model, the better alignment results. Any user can append or replace the acoustic models included in the “models” folder of the “resources” directory. Be aware that SPPAS only supports HTK-ASCII acoustic models, trained from 16 bits, 16000 Hz wave files.

The existing models can be improved if they are re-trained with more data. To get a better alignment result, any new data is then welcome: send an e-mail to the author to share your recordings and transcripts.

3.11.3 Support of a new language

The support of a new language in Alignment only consists in adding a new acoustic model of the appropriate format, in the appropriate directory, with the appropriate phone set.

The articulatory representations of phonemes are so similar across languages that phonemes can be considered as units which are independent from the underlying language (Schultz et al. 2001). In SPPAS package, 9 acoustic models of the same type - i.e. same HMMs definition and acoustic parameters, are already available so that the phoneme prototypes can be extracted and reused to create an initial model for a new language.

Any new model can also be trained by the author, as soon as enough data is available. It is difficult to estimate exactly the amount of data a given language requires. That is said, we can approximate the minimum as follow:

- 3 minutes altogether of various speakers, manually time-aligned at the phoneme level.
- 10 minutes altogether of various speakers, time-aligned at the ipus level with the enriched orthographic transcription.
- more data is good data.

3.11.4 Perform Alignment with the GUI

It is an annotation of STANDALONE type.

The Alignment process takes as input one or two files that strictly match the audio file name except for the extension and that “-phon” is appended for the first one and “-token” for the optional second one. For example, if the audio file name is “oriana1.wav”, the expected input file name is “oriana1-phon.xra” with phonetization and optionally “oriana1-token.xra” with text normalization, if .xra is the default extension for annotations.

The speech segmentation process provides one file with name “-palign” appended to its name, i.e. “oriana1-palign.xra” for the previous example. This file includes one or two tiers:

- “PhonAlign” is the segmentation at the phone level;

- “TokensAlign” is the segmentation at the word level (if a file with tokenization was found).

The following options are available to configure Alignment:

- choose the speech segmentation system. It can be either: julius, hvite or basic
- perform basic alignment if the aligner failed, instead such intervals are empty.
- remove working directory will keep only alignment result: it will remove working files. Working directory includes one wav file per unit and a set of text files per unit.
- create the PhnTokAlign will append another tier with intervals of the phonetization of each word.

To perform the annotation, click on the Alignment activation button, select the language and click on the “Configure...” blue text to fix options.

3.11.5 Perform Alignment with the CLI

`alignment.py` is the program to perform automatic speech segmentation of a given phonetized file.

Usage

```
alignment.py [files] [options]
```

Alignment: Time-alignment of speech audio with its corresponding transcription at the phone and token levels. Requires a Phonetization.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

<code>-i file</code>	Input wav file name.
<code>-p file</code>	Input file name with the phonetization.
<code>-t file</code>	Input file name with the tokenization.
<code>-o file</code>	Output file name with estimated alignments.

Files (auto mode):

<code>-I file</code>	Input transcription file name (append).
<code>-l lang</code>	Language code (iso8859-3). One of: cat cmn deu eng eng-cd fra ita jpn kor nan pcm pol por spa yue.
<code>-e .ext</code>	Output file extension. One of: .xra .TextGrid .eaf .csv .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff

Resources:

<code>-r model</code>	Directory of the acoustic model of the language of the text
<code>-R model</code>	Directory of the acoustic model of the mother language of the speaker (under development)

Options:

<code>--inputpattern INPUTPATTERN</code>	Input file pattern (phonetization) (default: -phon)
--	---

```
--inputoptpattern INPUTOPTPATTERN
                        Optional input file pattern (tokenization) (default:
                        -token)
--outputpattern OUTPUTPATTERN
                        Output file pattern (default: -palign)
--aligner ALIGNER       Speech automatic aligner system (julius, hvite,
                        basic): (default: julius)
--basic BASIC           Perform basic alignment if the aligner fails (default:
                        False)
--clean CLEAN           Remove working directory (default: True)
--activity ACTIVITY     Create the Activity tier (default: True)
--activityduration ACTIVITYDURATION
                        Create the ActivityDuration tier (default: False)
```

This program is part of SPPAS version 2.4. Copyright (C) 2011-2019 Brigitte Bigi. Contact the author at: contact@sppas.org

Example of use

```
python .\sppas\bin\alignment.py -I .\samples\samples-eng\oriana1.wav -l eng
2018-12-19 18:33:38,842 [INFO] Logging set up level=15
2018-12-19 18:33:38,844 [INFO] Options
2018-12-19 18:33:38,844 [INFO] ... activityduration: False
2018-12-19 18:33:38,845 [INFO] ... activity: True
2018-12-19 18:33:38,845 [INFO] ... aligner: julius
2018-12-19 18:33:38,845 [INFO] ... clean: True
2018-12-19 18:33:38,845 [INFO] ... basic: False
2018-12-19 18:33:38,845 [INFO] File oriana1.wav: Valid.
2018-12-19 18:33:38,845 [INFO] File oriana1-phon.xra: Valid.
2018-12-19 18:33:38,846 [INFO] File oriana1-token.xra: Valid.
2018-12-19 18:33:38,846 [WARNING] ... ... A file with name E:\bigi\Projets\sppas\samples\samp
2018-12-19 18:33:38,855 [INFO] ... Découpage en intervalles.
2018-12-19 18:33:38,901 [INFO] ... Intervalle numéro 1.
2018-12-19 18:33:38,904 [INFO] ... Intervalle numéro 2.
2018-12-19 18:33:38,908 [INFO] ... Intervalle numéro 3.
2018-12-19 18:33:38,913 [INFO] ... Intervalle numéro 4.
2018-12-19 18:33:38,917 [INFO] ... Intervalle numéro 5.
2018-12-19 18:33:38,921 [INFO] ... Intervalle numéro 6.
2018-12-19 18:33:38,926 [INFO] ... Intervalle numéro 7.
2018-12-19 18:33:38,928 [INFO] ... Fusion des alignements des intervalles.
2018-12-19 18:33:38,969 [INFO] ... Création de la tier des activités.
2018-12-19 18:33:38,993 [INFO] ... E:\bigi\Projets\sppas\samples\samples-eng\oriana1-palign.x
```

3.12 Activity

3.12.1 Overview

Activity tier represents speech activities, i.e. speech, silences, laughter, noises... It is based on the analysis of the time-aligned tokens.

3.12.2 Perform Activity with the GUI

It is an annotation of STANDALONE type.

The Activity process takes as input a file that strictly match the audio file name except for the extension and that “-palign” is appended. For example, if the audio file name is “oriana1.wav”, the expected input file name is “oriana1-palign.xra” if .xra is the default extension for annotations. This file must include time-aligned phonemes in a tier with name “PhonAlign”.

The annotation provides an annotated file with “-activity” appended to its name, i.e. “oriana1-activity.xra” for the previous example. This file is including 1 or 2 tiers: Activity, ActivityDuration.

To perform the annotation, click on the Activity activation button and click on the “Configure...” blue text to fix options.

3.12.3 Perform Alignment with the CLI

No CLI is available for this annotation.

3.13 RMS

3.13.1 Overview

The Root-Mean Square - RMS is a measure of the power in an audio signal. It is estimated from the amplitude values by: $\sqrt{\sum(S_i^2)/n}$.

RMS automatic annotation estimates the rms value on given intervals of an audio file. Empty intervals - i.e. intervals without labels, are ignored. By default, the RMS is estimated on a tier with name “PhonAlign” of an annotated file with pattern “-palign”. Both can be modified by configuring the annotations. The annotation provides an annotated file with “-rms” appended to its name. This file is including 3 tiers:

- RMS: indicates the RMS value estimated on each non-empty interval;
- RMS-values: indicates RMS values estimated every 10 ms in each interval;
- RMS-mean: indicates the mean of the previous values.

3.13.2 Perform RMS with the GUI

It is an annotation of STANDALONE type.

To perform the annotation, click on the RMS activation button and click on the “Configure...” blue text to fix options.

3.13.3 Perform RMS with the CLI

`rms.py` is the program to perform this annotation, either on a single given file (-i and -t) or on a set of files (-I).

3.14 Interval Values Analysis - IVA

3.14.1 Overview

The Interval Values Analysis - IVA is producing statistical information about a set of values in given intervals. IVA can for example estimate the mean/stddev values on given intervals (IPUs, ...) of a pitch file. Empty intervals - i.e. unlabelled intervals, are ignored and a list of tags to be ignored can be fixed.

By default, the IVA is estimated with the values of a PitchTier inside the intervals defined in a tier with name “TokensAlign” of a file with pattern “-palign”. If a list of separators is given, the intervals are created: an IVA segment is a set of consecutive annotations without separators. Default separators are “# + @ * dummy” in order to ignore silences, laughter items, noises and untranscribed speech. However, if no separator is given, IVA segments are matching the intervals of the given input tier. In the latter case, be aware that some file formats - including TextGrid, are not supporting holes: they create unlabelled intervals between the labelled ones.

Both tiernames and patterns can be modified by configuring the annotation. The annotation provides an annotated file with the “-iva” pattern. This file includes the tiers:

- IVA-Segments: defined intervals;
- IVA-Values: the values extracted for each segment;
- IVA-Occurrences: indicates the number of values of each segment;
- IVA-Total: indicates the sum of values of each segment;
- IVA-Mean: indicates the mean of values of each segment;
- IVA-Median: indicates the median of values of each segment;
- IVA-StdDev: indicates the standard deviation of values of each segment;
- IVA-Intercept: indicates the intercept value of the linear regression of values of each segment;
- IVA-Slope: indicates the slope value of the linear regression of values of each segment.

3.14.2 Perform IVA with the GUI

It is an annotation of STANDALONE type.

To perform the annotation, click on the IVA activation button and click on the “Configure...” blue text to fix options.

3.14.3 Perform IVA with the CLI

`iva.py` is the program to perform this annotation, either on a single given file (-i and -s) or on a set of files (-I).

3.15 Lexical Metric

3.15.1 Overview

The Lexical Metric is producing information about the number of occurrences and the rank of each eaccurrences of annotation labels.



Figure 3.8: Syllabification example

By default, the lexical metrics are estimated on a tier with name “TokensAlign” of a file with pattern “-palign”. If a list of separators is given, segments are created to estimate a number of occurrences. Default separators are “# + @ * dummy” in order to ignore silences, laughter items, noises and untranscribed speech.

Both the tiername and the pattern can be modified by configuring the annotation. The annotation provides an annotated file with the “-lexm” pattern. This file includes the tiers:

- LM-OccAnnInSegments: defined intervals with number of occurrences of annotations;
- LM-OccLabInSegments: defined intervals with number of occurrences of labels;
- LM-Occ: the number of occurrences of each label the annotation is representing;
- LM-Rank: the rank of each label the annotation is representing.

3.15.2 Perform Lexical Metric with the GUI

It is an annotation of STANDALONE type.

To perform the annotation, click on the Lexical Metric activation button and click on the “Configure...” blue text to fix options.

3.16 Syllabification

3.16.1 Overview

The syllabification of phonemes is performed with a rule-based system from time-aligned phonemes. This phoneme-to-syllable segmentation system is based on 2 main principles:

- a syllable contains a vowel, and only one;
- a pause is a syllable boundary.

These two principles focus the problem of the task of finding a syllabic boundary between two vowels. Phonemes were grouped into classes and rules are established to deal with these classes.

For each language, the automatic syllabification requires a configuration file to fix phonemes, classes and rules.

3.16.2 Adapt Syllabification

Any user can change the set of rules by editing and modifying the configuration file of a given language. Such files are located in the folder “syll” of the “resources” directory. Files are all with **UTF-8 encoding** and **“LF” for newline**.

At first, the list of phonemes and the class symbol associated with each of the phonemes are described as, for example:

- PHONCLASS e V
- PHONCLASS p P

Each association phoneme/class definition is made of 3 columns: the first one is the key-word PHONCLASS, the second is the phoneme symbol (like defined in the tier with the phonemes, commonly X-SAMPA), the last column is the class symbol. The constraints on this definition are that a class-symbol is only one upper-case character, and that the character X is forbidden, and the characters V and W are reserved for vowels.

The second part of the configuration file contains the rules. The first column is a keyword, the second one describes the classes between two vowels and the third column is the boundary location. The first column can be:

- GENRULE
- EXCRULE
- OTHRULE.

In the third column, a “0” means the boundary is just after the first vowel, “1” means the boundary is one phoneme after the first vowel, etc. Here are some examples of the file for French language:

- GENRULE VXV 0
- GENRULE VXXV 1
- EXCRULE VFLV 0
- EXCRULE VOLGV 0

Finally, to adapt the rules to specific situations that the rules failed to model, we introduced some phoneme sequences and the boundary definition. Specific rules contain only phonemes or the symbol “ANY” which means any phoneme. It consists of 7 columns: the first one is the key-word OTHRULE, the 5 following columns are a phoneme sequence where the boundary should be applied to the third one by the rules, the last column is the shift to apply to this boundary. In the following example:

```
OTHRULE ANY ANY p s k -2
```

More information are available in (Bigi et al. 2010).

3.16.3 Support of a new language

The support of a new language in this automatic syllabification only consists in adding a configuration file (see previous section). Fix properly the encoding (utf-8) and newlines (LF) of this file; then fix the name and extension of the file as follow:

- “syllConfig-” followed by language name with iso639-3 standard,
- with extension “.txt”.

3.16.4 Perform Syllabification with the GUI

It is an annotation of STANDALONE type.

The Syllabification process takes as input a file that strictly match the audio file name except for the extension and that “-palin” is appended. For example, if the audio file name is “oriana1.wav”, the expected input

file name is “oriana1-align.xra” if .xra is the default extension for annotations. This file must include time-aligned phonemes in a tier with name “PhonAlign”.

The annotation provides an annotated file with “-salign” appended to its name, i.e. “oriana1-salign.xra” for the previous example. This file is including 2 tiers: SyllAlign, SyllClassAlign.

To perform the annotation, click on the Syllabification activation button, select the language and click on the “Configure...” blue text to fix options.

3.16.5 Perform Syllabification with the CLI

syllabify.py is the program to perform automatic syllabification of a given file with time-aligned phones.

Usage

```
syllabify.py [files] [options]
```

Syllabification: Syllabification is based on a set of rules to convert phonemes into classes and to group them. Requires time-aligned phones.

optional arguments:

-h, --help	show this help message and exit
--quiet	Disable the verbosity
--log file	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

-i file	Input time-aligned phonemes file name.
-o file	Output file name with syllables.

Files (auto mode):

-I file	Input transcription file name (append).
-l lang	Language code (iso8859-3). One of: fra ita pol.
-e .ext	Output file extension. One of: .xra .TextGrid .eaf .csv .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff

Resources:

-r rules	Configuration file with syllabification rules
----------	---

Options:

--inputpattern INPUTPATTERN	Input file pattern (time-aligned phonemes) (default: -palign)
--outputpattern OUTPUTPATTERN	Output file pattern (default: -syll)
--usesphons USESPHONS	Syllabify inside the IPU intervals (default: True)
--usesintervals USESINTERVALS	Syllabify inside an interval tier (default: False)
--tiername TIERNAM	Tier name for such interval tier: (default: TokensAlign)
--createclasses CREATECLASSES	Create a tier with syllable classes (default: True)

This program is part of SPPAS version 2.4. Copyright (C) 2011–2019 Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

```
python .\sppas\bin\syllabify.py -i .\samples\samples-fra\F_F_B003-P8-palign.xra
-r .\resources\syll\syllConfig-fra.txt --quiet
SyllAlign
2.497101 2.717101 j-E-R
2.717101 2.997101 s-w-A/-R
...
19.412000 19.692000 P-L-V-P
19.692000 20.010000 P-V-L-P
```

All the following commands will produce the same result:

```
python .\sppas\bin\syllabify.py -I .\samples\samples-fra\F_F_B003-P8-palign.xra -l fra
python .\sppas\bin\syllabify.py -I .\samples\samples-fra\F_F_B003-P8.TextGrid -l fra
python .\sppas\bin\syllabify.py -I .\samples\samples-fra\F_F_B003-P8.wav -l fra
python .\sppas\bin\syllabify.py -I .\samples\samples-fra\F_F_B003-P8 -l fra
```

3.17 TGA - Time Groups Analyzer

3.17.1 Overview

TGA is originally available at <http://wwwhomes.uni-bielefeld.de/gibbon/TGA/>. It's a tool developed by Dafydd Gibbon, emeritus professor of English and General Linguistics at Bielefeld University.

*Dafydd Gibbon (2013). **TGA: a web tool for Time Group Analysis**, Tools and Resources for the Analysis of Speech Prosody, Aix-en-Provence, France, pp. 66-69.*

The original TGA is an online batch processing tool which provides a parametrised mapping from time-stamps in speech annotation files in various formats to a detailed analysis report with statistics and visualisations. TGA software calculates, inter alia, mean, median, rPVI, nPVI, slope and intercept functions within inter-pausal groups, provides visualizations of timing patterns, as well as correlations between these, and parses inter-pausal groups into hierarchies based on duration relations. Linear regression is selected mainly for the slope function, as a first approximation to examining acceleration and deceleration over large data sets.

The TGA online tool was designed to support phoneticians in basic statistical analysis of annotated speech data. In practice, the tool provides not only rapid analyses but also the ability to handle larger data sets than can be handled manually.

In addition to the original one, a second version of TGA was implemented in the AnnotationPro software:

*Katarzyna Klessa, Dafydd Gibbon (2014). **Annotation Pro + TGA: automation of speech timing analysis**, 9th International conference on Language Resources and Evaluation (LREC), Reykjavik (Iceland). pp. 1499-1505, ISBN: 978-2-9517408-8-4.*

The integrated Annotation Pro + TGA tool incorporates some TGA features and is intended to support the development of more robust and versatile timing models for a greater variety of data. The integration of TGA statistical and visualisation functions into Annotation Pro+TGA results in a powerful computational enhancement of the

existing AnnotationPro phonetic workbench, for supporting experimental analysis and modeling of speech timing.

So, what's the novelty into the third version implemented into SPPAS...

First of all, it has to be noticed that TGA is only partly implemented into SPPAS. The statistics analyses tool of SPPAS allows to estimates TGA within the SPPAS framework; and it results in the following advantages:

- it can read either TextGrid, csv, Elan, or any other file format supported by SPPAS,
- it can save TGA results in any of the annotation file supported by SPPAS,
- it estimates the two versions of the linear regression estimators: the original one and the one implemented into AnnotationPro:
 1. in the original TGA, the x-axis is based on positions of syllables,
 2. in the AnnotationPro+TGA, the x-axis is based on time-stamps.

3.17.2 Result of TGA into SPPAS

The annotation provides an annotated file with “-tga” appended to its name, i.e. “oriana1-tga.xra” for the example. This file is including 10 tiers:

1. TGA-TimeGroups: intervals with the time groups
2. TGA-TimeSegments: same intervals, indicate the syllables separated by whitespace
3. TGA-Occurrences: same intervals, indicate the number of syllables
4. TGA-Total: same intervals, indicate interval duration
5. TGA-Mean: same intervals, indicate mean duration of syllables
6. TGA-Median: same intervals, indicate median duration of syllables
7. TGA-Stdev: same intervals, indicate stdev of duration of syllables
8. TGA-nPVI: same intervals, indicate nPVI of syllables
9. TGA-Intercept: same intervals, indicate the intercept
10. TGA-Slope: same intervals, indicate the slope

Both tiers 9 and 10 can be estimated in 2 ways (so 2 more tiers can be generated).

3.17.3 Perform TAG with the GUI

It is an annotation of STANDALONE type.

The TGA process takes as input a file that strictly match the audio file name except for the extension and that “-salgn” is appended. For example, if the audio file name is “oriana1.wav”, the expected input file name is “oriana1-salgn.xra” if .xra is the default extension for annotations. This file must include time-aligned syllables in a tier with name “SyllAlign”.

To perform the annotation, click on the TGA activation button and click on the “Configure...” blue text to fix options.

3.17.4 Perform TGA with the CLI

`tga.py` is the program to perform TGA of a given file with time-aligned syllables.

Usage

`tga.py` [files] [options]

TimeGroupAnalysis: Proposed by D. Gibbon, Time Group Analyzer calculates mean, median, nPVI, slope and intercept functions within inter-pausal groups. Requires time aligned syllables.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

<code>-i file</code>	An input time-aligned syllables file.
<code>-o file</code>	Output file name with TGA.

Files (auto mode):

<code>-I file</code>	Input time-aligned syllables file (append).
<code>-e .ext</code>	Output file extension. One of: .xra .TextGrid .eaf .csv .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff

Options:

<code>--original ORIGINAL</code>	Use the original estimation of intercept and slope (default: False)
<code>--annotationpro ANNOTATIONPRO</code>	Use the estimation of intercept and slope proposed in AnnotationPro (default: True)
<code>--tg_prefix_label TG_PREFIX_LABEL</code>	Prefix of each time group label: (default: tg_)
<code>--with_radius WITH_RADIUS</code>	Duration estimation: Use 0 to estimate syllable durations with midpoint values, use -1 for Radius-, or 1 for Radius+. (default: 0)

This program is part of SPPAS version 2.0. Copyright (C) 2011-2019 Brigitte Bigi. Contact the author at: contact@sppas.org

Example of use

```
python .\sppas\bin\tga.py -i .\samples\samples-fra\F_F_B003-P8-syll.xra
2018-12-20 08:35:21,219 [INFO] Logging set up level=15
TGA-TimeGroups
2.497101 5.683888 tg_1
5.743603 8.460596 tg_2
9.145000 11.948531 tg_3
12.494000 13.704000 tg_4
13.784000 15.036000 tg_5
16.602000 20.010000 tg_6
```

```
TGA-TimeSegments
...
13.784000 15.036000 -0.03063
16.602000 20.010000 0.00468
```

Other commands:

```
python .\sppas\bin\tga.py -I .\samples\samples-fra\F_F_B003-P8-syll.xra
python .\sppas\bin\tga.py -I .\samples\samples-fra\F_F_B003-P8.TextGrid
python .\sppas\bin\tga.py -I .\samples\samples-fra\F_F_B003-P8.wav
```

3.18 Stop words

Create a tier with True/False indicating if a token is a stop-word or not.

3.19 Self-Repetitions

3.19.1 Overview

This automatic detection focus on word self-repetitions which can be exact repetitions (named strict echos) or repetitions with variations (named non-strict echos). The system is based only on lexical criteria. The algorithm is focusing on the detection of the source.

This system can use a list of stop-words of a given language. This is a list of very frequent words like adjectives, pronouns, etc. Obviously, the result of the automatic detection is significantly better if such list of stop-words is available.

Optionnally, SPPAS can add new stop-words in the list: they are deduced from the given data. These new entries in the stop-list are then different for each file (Bigi et al. 2014).

The annotation provides one annotated file with 2 to 4 tiers:

1. TokenStrain: if a replacement file was available, it's the entry used by the system
2. StopWord: if a stop-list was used, it indicates if the token is a stop-word (True or False)
3. SR-Sources: tags of the annotations are prefixed by "S" followed an index
4. SR-Repetitions: tags of the annotations are prefixed by "R" followed an index

3.19.2 Adapt to a new language

The list of stop-words of a given language must be located in the "vocab" folder of the "resources" directory with ".stp" extension. This file is with [UTF-8 encoding](#) and ["LF" for newline](#).

3.19.3 Perform Self-Repetitions with the GUI

It is an annotation of STANDALONE type.

The automatic annotation takes as input a file with (at least) one tier containing the time-aligned tokens. The annotation provides one annotated file with 2 tiers: Sources and Repetitions.

Click on the Self-Repetitions activation button, select the language and click on the "Configure..." blue text to fix options.

3.19.4 Perform SelfRepetitions with the CLI

`selfrepetition.py` is the program to perform automatic detection of self-repetitions.

Usage

`selfrepetition.py` [files] [options]

Self-repetitions: Self-repetitions searches for sources and echos of a speaker. Requires time-aligned tokens.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

<code>-i file</code>	Input time-aligned tokens file name.
<code>-o file</code>	Output file name with syllables.

Files (auto mode):

<code>-I file</code>	Input transcription file name (append).
<code>-e .ext</code>	Output file extension. One of: <code>.xra</code> <code>.TextGrid</code> <code>.eaf</code> <code>.csv</code> <code>.mrk</code> <code>.txt</code> <code>.stm</code> <code>.ctm</code> <code>.lab</code> <code>.mlf</code> <code>.sub</code> <code>.srt</code> <code>.antx</code> <code>.arff</code> <code>.xrff</code>

Resources:

<code>-r file</code>	List of stop-words
----------------------	--------------------

Options:

<code>--inputpattern INPUTPATTERN</code>	Input file pattern (time-aligned words or lemmas) (default: <code>-palign</code>)
<code>--outputpattern OUTPUTPATTERN</code>	Output file pattern (default: <code>-srepet</code>)
<code>--span SPAN</code>	Span window length in number of IPUs (default: 3)
<code>--stopwords STOPWORDS</code>	Add stop-words estimated from the given data (default: True)
<code>--alpha ALPHA</code>	Coefficient to add data-specific stop-words (default: 0.5)

This program is part of SPPAS version 3.0. Copyright (C) 2011–2020 Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

```
python .\sppas\bin\selfrepetition.py -i .\samples\samples-fra\F_F_B003-P8-palign.xra
-r .\resources\vocab\fra.stp
python .\sppas\bin\selfrepetition.py -I .\samples\samples-fra\F_F_B003-P8.wav -l fra
```

3.20 Other-Repetitions

3.20.1 Overview

This automatic detection focus on other-repetitions, which can be either exact repetitions (named strict echos) or repetitions with variations (named non-strict echos). The system is based only on lexical criteria (Bigi et al. 2014). Notice that the algorithm is focusing on the detection of the source.

This system can use a list of stop-words of a given language. This is a list of very frequent words like adjectives, pronouns, etc. Obviously, the result of the automatic detection is significantly better if such list of stop-words is available.

Optionnaly, SPPAS can add new stop-words in the list: they are deduced from the given data. These new entries in the stop-list are then different for each file (see Bigi et al. 2014).

The detection of the ORs is performed in a span window of N IPU; by default, N is fixed to 5. It means that if a repetition is after these N IPU, it won't be detected. Technically, it also means that SPPAS needs to identify the boundaries of the IPU from the time-aligned tokens: the tier must indicate the silences with the '#' symbol.

A file with the following tiers will be created:

- OR-Source: intervals with the number of the sources
- OR-SrcStrain: intervals with the tokens of the sources
- OR-SrcLen: intervals with the number of tokens in the source
- OR-SrcType: intervals with the type of the echos of the sources
- OR-Echo: intervals with the number of the echos

3.20.2 Adapt to a language and support of a new one

This system can use a list of stop-words of a given language. This is a list of very frequent words like adjectives, pronouns, etc. Obviously, the result of the automatic detection is significantly better if such list of stop-words is available. It must be located in the "vocab" folder of the "resources" directory with ".stp" extension. This file is with [UTF-8 encoding](#) and ["LF" for newline](#).

3.20.3 Perform Other-Repetitions with the GUI

It is an annotation of INTERACTION type.

The automatic annotation takes as input a file with (at least) one tier containing the time-aligned tokens of the main speaker, and another file/tier with tokens of the interlocutor. The annotation provides one annotated file with 2 tiers: Sources and Repetitions.

Click on the Other-Repetitions activation button, select the language and click on the "Configure..." blue text to fix options.

3.20.4 Perform Other-Repetitions with the CLI

```
usage: otherrepetition.py -r stopwords [files] [options]
```

Files:

<code>-i file</code>	Input file name with time-aligned tokens of the main speaker.
<code>-s file</code>	Input file name with time-aligned tokens of the echoing speaker
<code>-o file</code>	Output file name with ORs.

Options:

<code>--inputpattern INPUTPATTERN</code>	Input file pattern (time-aligned words or lemmas) (default: <code>-palign</code>)
<code>--outputpattern OUTPUTPATTERN</code>	Output file pattern (default: <code>-orepet</code>)
<code>--span SPAN</code>	Span window length in number of IPUs (default: 3)
<code>--stopwords STOPWORDS</code>	Add stop-words estimated from the given data (default: <code>True</code>)
<code>--alpha ALPHA</code>	Coefficient to add data-specific stop-words (default: 0.5)

3.21 Re-Occurrences

This annotation is searching for re-occurrences of an annotation of a speaker in the next N annotations of the interlocutor. It is originally used for gestures in (M. Karpinski et al. 2018).

Maciej Karpinski, Katarzyna Klessa Methods, Tools and Techniques for Multimodal Analysis of Accommodation in Intercultural Communication CMST 24(1) 29–41 (2018), DOI:10.12921/cmst.2018.0000006

3.21.1 Perform Re-Occurrences with the GUI

The automatic annotation takes as input any annotated file with (at least) one tier, and another file+tier of the interlocutor. The annotation provides one annotated file with 2 tiers: Sources and Repetitions.

Click on the Re-Occurrences activation button, and click on the “Configure...” blue text to fix options.

3.21.2 Perform Re-Occurrences with the CLI

usage: `reoccurrences.py` [files] [options]

Files:

<code>-i file</code>	Input file name with time-aligned annotations of the main speaker.
<code>-s file</code>	Input file name with time-aligned annotations of the interlocutor
<code>-o file</code>	Output file name with re-occurrences.

Options:

```
--inputpattern INPUTPATTERN
                        Input file pattern (default: )
--outputpattern OUTPUTPATTERN
                        Output file pattern (default: -reocc)
--tiername TIERNAME     Tier to search for re-occurrences (default: )
--span SPAN             Span window length in number of annotations (default:
                        10)
```

This program is part of SPPAS version 2.4. Copyright (C) 2011-2019 Brigitte Bigi. Contact the author at: contact@sppas.org

3.22 Momel (modelling melody)

Momel is an algorithm for the automatic modeling of fundamental frequency (F0) curves using a technique called asymmetric modal quadratic regression.

This technique makes it possible by an appropriate choice of parameters to factor an F0 curve into two components:

- a macro-prosodic component represented by a quadratic spline function defined by a sequence of target points < ms, hz >.
- a micro-prosodic component represented by the ratio of each point on the F0 curve to the corresponding point on the quadratic spline function.

For details, see the following reference:

Daniel Hirst and Robert Espesser (1993). *Automatic modelling of fundamental frequency using a quadratic spline function*. Travaux de l'Institut de Phonétique d'Aix. vol. 15, pages 71-85.

The SPPAS implementation of Momel requires a file with the F0 values **sampled at 10 ms**. Two file formats are supported:

- “.PitchTier”, from Praat.
- “.hz”, from any tool. It is a file with one F0 value per line.

The following options can be fixed:

- Window length used in the “cible” method
- F0 threshold: Maximum F0 value
- F0 ceiling: Minimum F0 value
- Maximum error: Acceptable ratio between two F0 values
- Window length used in the “reduc” method
- Minimal distance
- Minimal frequency ratio
- Eliminate glitch option: Filter f0 values before ‘cible’

3.22.1 Perform Momel with the GUI

It is an annotation of STANDALONE type.

Click on the Momel activation button then click on the “Configure...” blue text to fix options.

3.22.2 Perform Momel with the CLI

`momel.py` is the program to perform Momel annotation of a given file with F0 values sampled at 10ms.

Usage

`momel.py [files] [options]`

Momel: Proposed by D. Hirst and R. Espesser, Momel - Modelling of fundamental frequency (F0) curves is using a technique called assymetric modal quaratic regression. Requires pitch values.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report (default: None)

Files (manual mode):

<code>-i file</code>	Input file name (extension: .hz or .PitchTier)
<code>-o file</code>	Output file name (default: stdout)

Files (auto mode):

<code>-I file</code>	Input file name with pitch (append).
<code>-e .ext</code>	Output file extension. One of: .xra .TextGrid .eaf .csv .mrk .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff

Options:

<code>--outputpattern OUTPUTPATTERN</code>	Output file pattern (default: -momel)
<code>--win1 WIN1</code>	Target window length (default: 30)
<code>--lo LO</code>	F0 threshold (default: 50)
<code>--hi HI</code>	F0 ceiling (default: 600)
<code>--maxerr MAXERR</code>	Maximum error (default: 1.04)
<code>--win2 WIN2</code>	Reduce window length (default: 20)
<code>--mind MIND</code>	Minimal distance (default: 5)
<code>--minr MINR</code>	Minimal frequency ratio (default: 0.05)

This program is part of SPPAS version 2.4. Copyright (C) 2011-2019 Brigitte Bigi. Contact the author at: contact@sppas.rg

Examples of use

```
python .\sppas\bin\momel.py -i .\samples\samples-eng\ENG_M15_ENG_T02.PitchTier
2018-12-19 15:44:00,437 [INFO] Logging set up level=15
2018-12-19 15:44:00,674 [INFO] ... .. 41 anchors found.
1.301629 109.285503
1.534887 126.157058
```

```
1.639614 143.657446
1.969234 102.911464
2.155284 98.550759
2.354162 108.250869
2.595364 87.005994
2.749773 83.577924
2.933222 90.218382
3.356651 119.709142
3.502254 104.104568
3.707747 132.055286
4.000578 96.262109
4.141915 93.741407
4.383332 123.996736
4.702203 89.152708
4.987086 101.561180
5.283864 87.499710
5.538984 92.399690
5.707147 95.411586
5.906895 87.081095
6.705373 121.396919
7.052992 130.821479
7.218415 120.917642
7.670083 101.867028
7.841935 109.094053
8.124574 90.763267
8.455182 114.261067
8.746016 93.704705
9.575359 101.108444
9.996245 122.488120
10.265663 105.244429
10.576394 94.875460
11.730570 99.698799
12.083323 124.002313
12.411790 108.563104
12.707442 101.928297
12.963805 113.980850
13.443483 90.782781
13.921939 90.824376
14.377324 60.126506
```

Apply Momel on all files of a given folder:

```
python .\sppas\bin\momel.py -I .\samples\samples-eng
```

3.23 INTSINT: Encoding of F0 anchor points

INTSINT assumes that pitch patterns can be adequately described using a limited set of tonal symbols, T,M,B,H,S,L,U,D (standing for : Top, Mid, Bottom, Higher, Same, Lower, Up-stepped, Down-stepped respectively) each one of which characterises a point on the fundamental frequency curve.

The rationale behind the INTSINT system is that the F0 values of pitch targets are programmed in one of two ways : either as absolute tones T, M, B which are assumed to refer to the speaker's overall pitch range

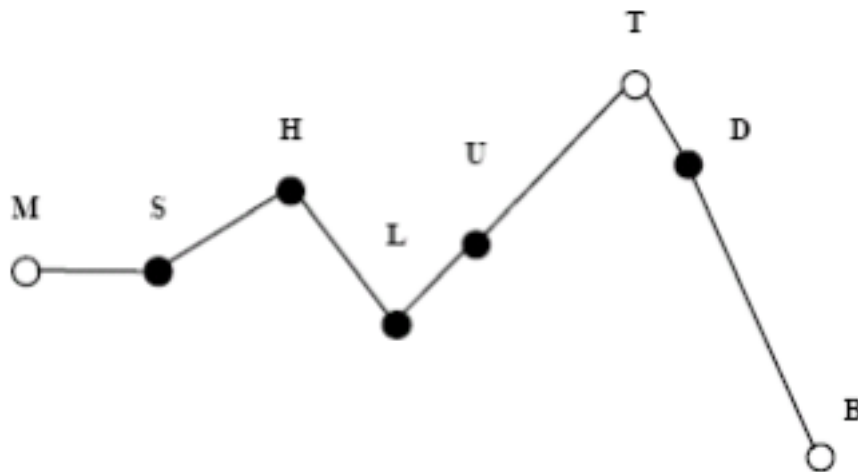
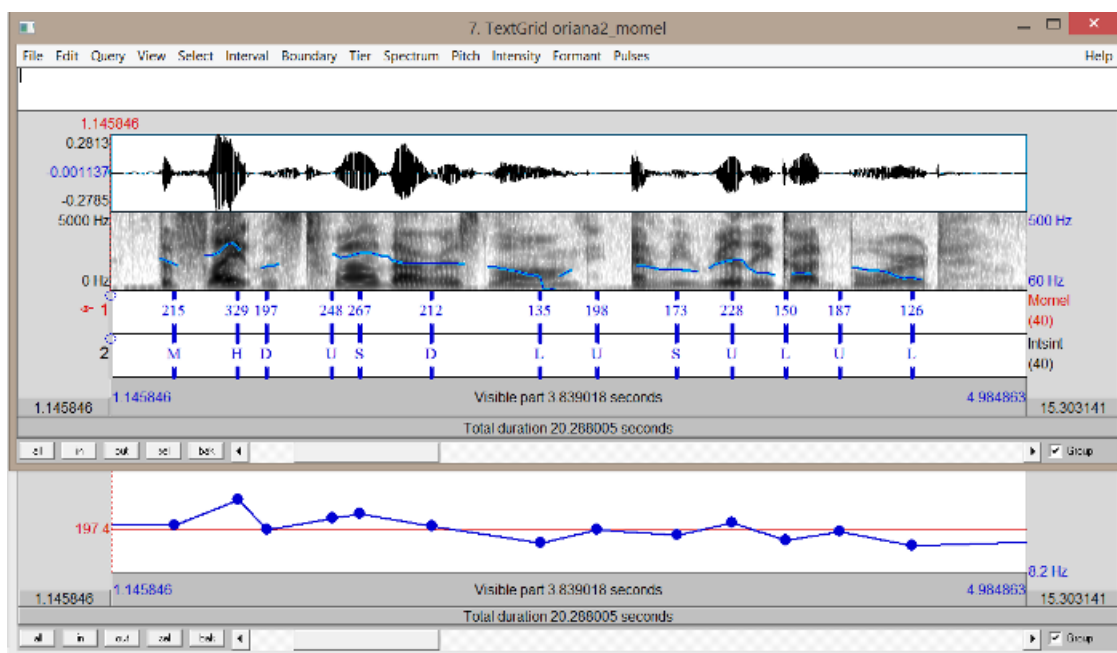


Figure 3.9: INTSINT example

(within the current Intonation Unit), or as relative tones H, S, L, U, D assumed to refer only to the value of the preceding target point.

The rationale behind the INTSINT system is that the F0 values of pitch targets are programmed in one of two ways : either as absolute tones T, M, B which are assumed to refer to the speaker's overall pitch range (within the current Intonation Unit), or as relative tones H, S, L, U, D assumed to refer only to the value of the preceding target point.

A distinction is made between non-iterative H, S, L and iterative U, D relative tones since in a number of descriptions it appears that iterative raising or lowering uses a smaller F0 interval than non-iterative raising or lowering. It is further assumed that the tone S has no iterative equivalent since there would be no means of deciding where intermediate tones are located.



D.-J. Hirst (2011). *The analysis by synthesis of speech melody: from data to models*, Journal of Speech Sciences, vol. 1(1), pages 55-83.

3.23.1 Perform INTSINT with the GUI

It is an annotation of STANDALONE type.

Click on the INTSINT activation button and click on the “Configure...” blue text to fix options.

3.23.2 Perform INTSINT with the CLI

`intsint.py` is the program to perform INTSINT annotation of a given file with momel anchors.

Usage

```
intsint.py [files] [options]
```

INTSINT: INternational Transcription System for INTonation codes the intonation of an utterance by means of an alphabet of 8 discrete symbols. Requires Momel targets.

optional arguments:

```
-h, --help  show this help message and exit
--quiet     Disable the verbosity
--log file  File name for a Procedure Outcome Report (default: None)
```

Files (manual mode):

```
-i file      Input file name with anchors.
-o file      Output file name (default: stdout)
```

Files (auto mode):

```
-I file      Input file name with anchors (append).
-e .ext      Output file extension. One of: .xra .TextGrid .eaf .csv .mrk
              .txt .stm .ctm .lab .mlf .sub .srt .antx .arff .xrff
```

Options:

```
--inputpattern INPUTPATTERN
                        Input file pattern (momel anchors) (default: -momel)
--outputpattern OUTPUTPATTERN
                        Output file pattern (default: -intsint)
```

This program is part of SPPAS version 2.4. Copyright (C) 2011-2019 Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

Apply INTSINT on a single file and print the result on the standard output:

```
python .\sppas\bin\intsint.py -i .\samples\samples-eng\ENG_M15_ENG_T02-momel.xra --quiet
1.301629 M
1.534887 U
1.639614 H
1.969234 L
2.155284 S
2.354162 U
```

2.595364 L
2.749773 S
2.933222 S
3.356651 H
3.502254 D
3.707747 H
4.000578 L
4.141915 S
4.383332 H
4.702203 L
4.987086 U
5.283864 L
5.538984 U
5.707147 D
5.906895 S
6.705373 M
7.052992 U
7.218415 S
7.670083 D
7.841935 S
8.124574 D
8.455182 U
8.746016 D
9.575359 M
9.996245 U
10.265663 D
10.576394 D
11.730570 M
12.083323 U
12.411790 D
12.707442 S
12.963805 U
13.443483 L
13.921939 S
14.377324 B

Apply INTSINT in auto mode:

```
python .\sppas\bin\intsint.py -I .\samples\samples-eng\ENG_M15_ENG_T02.wav  
python .\sppas\bin\intsint.py -I .\samples\samples-eng\ENG_M15_ENG_T02.PitchTier  
python .\sppas\bin\intsint.py -I .\samples\samples-eng\ENG_M15_ENG_T02-momel.xra
```

3.24 Face Detection

3.24.1 Overview

FaceDetection annotation allows to search for coordinates of faces in an image or in all images of a video. It requires both to enable the “video” feature in the setup to install the external libraries ‘numpy’ and ‘opencv-contrib’ and to check “facedetect” in the list of annotations to install.

Thanks to the opencv library, SPPAS is able to use two different systems and to combine their results:

1. an Artificial Neural Network (DNN);

2. an Haar Cascade Classifier (HCC).

The linguistic resources of this annotation include two DNN models and three models for HCC, including two frontal-face models and a profile-face one. By default, SPPAS launches two detectors only, i.e. 1 DNN and 1 HCC, and it combines their results. This annotation is about 2.5x real time. Even if it can increase the quality of the final result, other models are not used by default because the detection is very slow: 15x real time to use all 5 models. It is possible to use them by modifying the configuration file of the annotation, at your own risk: in the folder `sppas/etc`, copy the file `facedetect.json` into a backup file and rename the file `facedetect-extra.json` into `facedetect.json`.

3.24.2 Result of Face Detection

There are several output files that can be created:

- a copy of the image/video with all the detected faces surrounded by a square indicating a confidence score;
- as many cropped image files as the number of detected faces;
- a CSV file with coordinates and confidence score of each detected face.

There's also the possibility to consider the portrait - i.e. the face scaled by 2.1, instead of the face for all of these files.

3.24.3 Perform Face Detection with the GUI

It is a STANDALONE annotation.

The Face Detection process takes as input an image file and/or a video. To perform the annotation, click on the FaceDetection activation button and click on the "Configure..." blue text to fix options.

3.24.4 Perform Face Detection with the CLI

`facedetection.py` is the program to perform Face Detection annotation of a given media file.

Usage

```
usage: facedetection.py [files] [options]
```

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report (default: None)

Files:

<code>-i file</code>	Input image.
<code>-o file</code>	Output base name.
<code>-I file</code>	Input file name (append).
<code>-r model</code>	Model base name (.caffemodel or .xml models as wishes)
<code>-e .ext</code>	Output file extension (image or video)

Options:

```
--inputpattern INPUTPATTERN          Input file pattern (default: )
--outputpattern OUTPUTPATTERN         Output file pattern (default: -face)
--nbest NBEST                         Number of faces to select among those
                                     detected (0=auto) (default: 0)
--score SCORE                         Minimum confidence score to select detected
                                     faces (default: 0.2)
--portrait PORTRAIT                  Consider the portrait instead of the face in
                                     outputs (default: False)
--csv CSV                             Save coordinates of detected faces in a CSV
                                     file (default: False)
--tag TAG                             Surround the detected faces in the output
                                     image (default: True)
--crop CROP                           Save detected faces in cropped images
                                     (default: False)
--width WIDTH                         Resize all the cropped images to a fixed
                                     width (0=no) (default: 0)
--height HEIGHT                       Resize all the cropped images to a fixed
                                     height (0=no) (default: 0)
```

This program is part of SPPAS version 3.6. Copyright (C) 2011-2021
Brigitte Bigi. Contact the author at: contact@sppas.org

Examples of use

Example 1: CLI and GUI are both working in the same way. The results of the 2 pre-defined systems are combined.

```
python3 ./sppas/bin/facedetection.py -I ./samples/faces/BrigitteBigi_Aix2020.png --tag=True --crop
[INFO] Logging redirected to StreamHandler (level=0).
[INFO] SPPAS version 3.5
[INFO] Copyright (C) 2011-2021 Brigitte Bigi
[INFO] Web site: http://www.sppas.org/
[INFO] Contact: Brigitte Bigi (contact@sppas.org)
[INFO] * * * Annotation step 0 * * *
[INFO] Number of files to process: 1
[INFO] Options:
[INFO] ... inputpattern:
[INFO] ... outputpattern: -face
[INFO] ... nbest: 0
[INFO] ... score: 0.2
[INFO] ... portrait: True
[INFO] ... csv: True
[INFO] ... tag: True
[INFO] ... crop: True
[INFO] ... width: 0
[INFO] ... height: 0
[INFO] File BrigitteBigi_Aix2020.png: Valid.
[INFO] ... 3 faces found.
[INFO] ... ./samples/faces/BrigitteBigi_Aix2020-face.jpg
```

It creates the following 5 files in the samples/faces folder:

- BrigitteBigi_Aix2020-face.jpg with the 3 detected faces surrounded by a square and indicating the detection score
- BrigitteBigi_Aix2020_1-face.jpg: image of the face with the highest score
- BrigitteBigi_Aix2020_2-face.jpg: image of the face with the medium score
- BrigitteBigi_Aix2020_3-face.jpg: image of the face with the worse score
- BrigitteBigi_Aix2020-face.csv contains 3 lines with coordinates in columns 3-7 with (x,y) and (w,h) then the confidence score in column 8 ranging [0., 1.]

Notice that the image contains 3 faces and their positions are properly found.

Example 2: Only one system is used

```
python3 ./sppas/bin/facedetection.py -i ./samples/faces/BrigitteBigi_Aix2020.png
-r ./resources/faces/res10_300x300_ssd_iter_140000.caffemodel
--tag=True
-o ./samples/faces/BrigitteBigi_Aix2020-dnnface.png
[INFO] ... 4 faces found.
```

Example 3: Two systems are used

```
python3 ./sppas/bin/facedetection.py -i ./samples/faces/BrigitteBigi_Aix2020.png
-r ./resources/faces/haarcascade_frontalface_alt.xml
-r ./resources/faces/haarcascade_profileface.xml
--tag=True
-o ./samples/faces/BrigitteBigi_Aix2020-haarface.png
[INFO] ... 5 faces found.
```

3.25 Face Identity

3.25.1 Overview

Face Identity automatic annotation assigns a person identity to detected faces of a video. It takes as input a video and a CSV file with coordinates of the detected faces. It produces a CSV file with coordinates of the identified faces. Assigned persons names are “id-00x”. Obviously, the CSV file can be edited and such names can be changed *a posteriori*.

This annotation requires to enable the “video” feature in the setup, so it will install the external python libraries ‘numpy’ and ‘opencv-contrib’.

No external resources are needed.

3.25.2 Perform annotation with the GUI

It is a STANDALONE annotation.

The Face Identity process takes as input a video file. To perform the annotation, click on the Face Identity activation button and click on the “Configure...” blue text to fix options.

3.25.3 Perform with the CLI

`faceidentity.py` is the program to perform Face Identity annotation of a given video file, if the corresponding CSV file with detected faces is existing.

Usage

```
usage: faceidentity.py [files] [options]
```

optional arguments:

```
-h, --help            show this help message and exit
--quiet              Disable the verbosity
--log file           File name for a Procedure Outcome Report (default: None)
```

Files:

```
-i file              Input video.
-c file              Input CSV file with face coordinates and sights.
-o file              Output base name.
-I file              Input file name (append).
-e .ext              Output file extension. One of: .mp4 .avi .mkv
```

Options:

```
--inputpattern INPUTPATTERN
--inputoptpattern INPUTOPTPATTERN (default: -face)
--outputpattern OUTPUTPATTERN (default: -ident)
```

3.26 Face Landmarks

3.26.1 Overview

SPPAS is using the OpenCV's facial landmark API called `Facemark`. It includes three different implementations of landmark detection based on three different papers:

- `FacemarkKazemi`: This implementation is based on a paper titled "One Millisecond Face Alignment with an Ensemble of Regression Trees" by V.Kazemi and J. Sullivan published in CVPR 2014.
- `FacemarkAAM`: This implementation uses an Active Appearance Model and is based on an the paper titled "Optimization problems for fast AAM fitting in-the-wild" by G. Tzimiropoulos and M. Pantic, published in ICCV 2013.
- `FacemarkLBF`: This implementation is based a paper titled "Face alignment at 3000 fps via regressing local binary features" by S. Ren published in CVPR 2014.

The fundamental concept is that any person will have 68 particular points on the face (called sights). SPPAS is able to launch several of them and to combine their results in a single and hopefully better one.

This annotation requires both to enable the "video" feature in the setup in order to install the external libraries 'numpy' and 'opencv-contrib', and to check "facemark" in the list of annotations to be installed. Two different models will be downloaded and used: a Kazemi one and a LBF one.

3.26.2 Perform annotation with the GUI

It is a STANDALONE annotation.

The Face Sights process takes as input an image file and/or a video. To perform the annotation, click on the Face Sights activation button and click on the “Configure...” blue text to fix options.

3.26.3 Perform with the CLI

```
usage: facesights.py [files] [options]
```

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--quiet</code>	Disable the verbosity
<code>--log file</code>	File name for a Procedure Outcome Report

Files:

<code>-i file</code>	Input image.
<code>-o file</code>	Output base name.
<code>-I file</code>	Input file name (append).
<code>-r model</code>	Landmark model name (Kazemi, LBF or AAM)
<code>-R model</code>	FaceDetection model name
<code>-e .ext</code>	Output file extension.

Options:

<code>--inputpattern INPUTPATTERN</code>	
<code>--inputoptpattern INPUTOPTPATTERN</code>	(default: -face)
<code>--outputpattern OUTPUTPATTERN</code>	(default: -sights)

3.27 Cued speech - LPC

3.27.1 Overview

Speech reading or lip reading requires watching the lips of a speaker and is used for the understanding of the spoken sounds. However, various sounds have the same lips movement which implies a lot of ambiguity. In 1966, R. Orin Cornett invented the Cued Speech, a visual system of communication. It adds information about the pronounced sounds that are not visible on the lips.

Thanks to this code, speech reading is encouraged since the Cued Speech (CS) keys match all of the spoken phonemes but phonemes with the same movement have different keys. Actually, from both the hand position on the face (representing vowels) and handshapes, known as cues (representing consonants), CV syllables can be represented. So, a single CV syllable will be generated or decoded through both the lips position and the key of the hand.

LPC is the French acronym for “Langue Parlée Complétée”.

The conversion of phonemes into keys of CS is performed using a rule-based system. This RBS phoneme-to-key segmentation system is based on the only principle that a key is always of the form CV.

This annotation requires both to enable the “video” feature in the setup to install the external libraries ‘numpy’ and ‘opencv-contrib’ and to check “lpc” in the list of annotations.

3.27.2 Perform annotation with the GUI

It is a STANDALONE annotation.

The annotation process takes as input a “-palign” file and optionally a video. To perform the annotation, click on its activation button and click on the “Configure...” blue text to fix options.

3.27.3 Perform with the CLI

```
usage: cuedspeech.py [files] [options]
```

Convert files

4.1 Interoperability and compatibility: an introduction

The conversion of a file to another file is the process of changing the form of the presentation of the data, and not the data itself. Every time, when data file is to be used, they must be converted to a readable format for the next application. A data conversion is normally an automated process to some extent. SPPAS provide the possibility to automatically import and export the work done on some various file formats from a wide range of other software tools. For the users, the visible change will be only a different file extension but for software it is the difference between understanding of the contents of the file and the inability to read it.

The conversion of file formats is then a difficult task and it can imply that some data are left. Representing annotated data in SPPAS is of crucial importance for its automatic annotations, its analysis of annotations and for the software to be able to automatically annotate and analyze any kind of data files. SPPAS then includes an original and generic enough annotation representation framework. This framework for annotations is very rich and contain several information like alternative labels or alternative localizations of annotations, a tier hierarchy, controlled vocabularies, etc. A native format named XRA was then developed to fit in such data representation. The physical level of representation of XRA obviously makes use of XML, XML-related standards and stand-off annotation. Due to an intuitive naming convention, XRA documents are human readable as far as possible within the limits of XML.

4.2 SPPAS conversion method

In the scope of the compatibility between SPPAS data and annotated data from other software tools or programs, SPPAS is able to open/save and import/export several file formats.

SPPAS makes use of its internal data representation to convert files. A conversion then consists of two steps:

1. the incoming file is loaded and mapped to the SPPAS data framework;
2. such data is saved to the expected format.

These two steps are applied whatever the organization structure of annotations in the original or in the destination file format. This process allows SPPAS to import from and export to a variety of file formats. This is

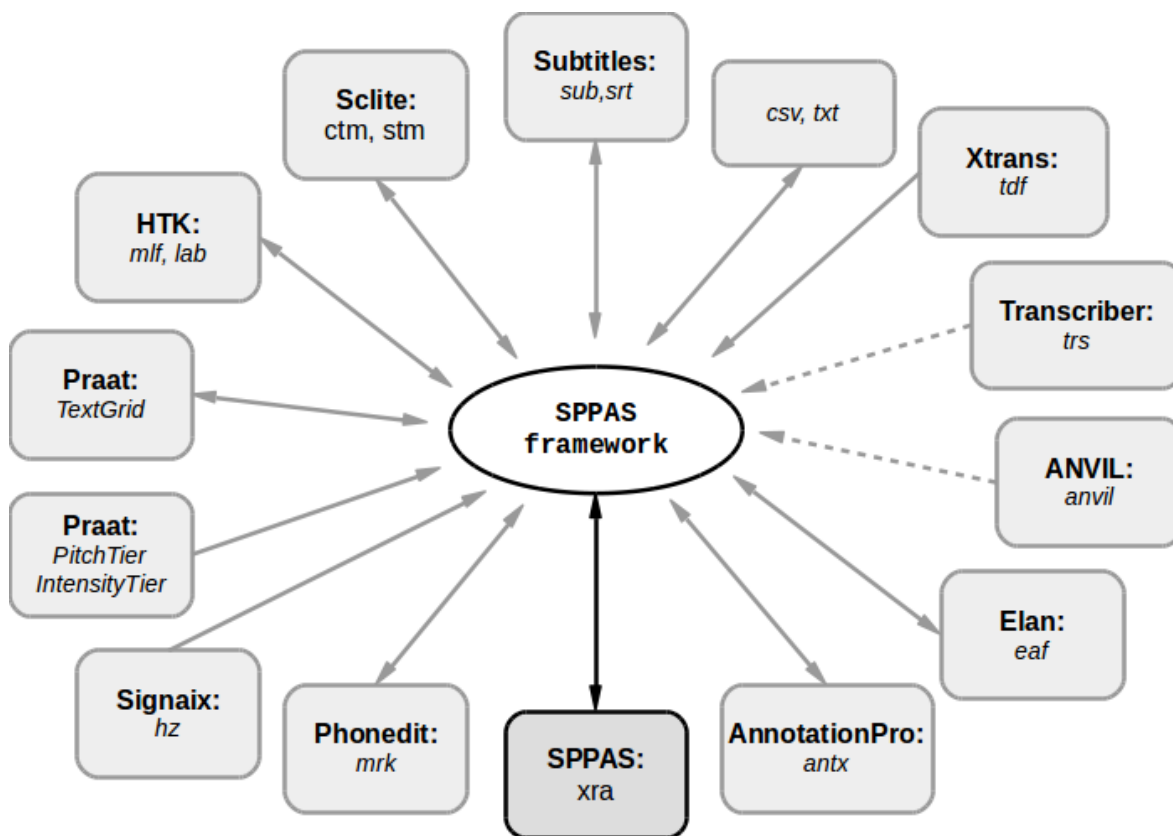


Figure 4.1: SPPAS conversion method and formats

illustrated by the next Figure which includes the list of file formats and the corresponding software that are supported. Arrows are representing the following actions:

- import from, represented by a continued line with an arrow from the file format to the SPPAS framework;
- partially import from, represented by a dash line with an arrow from the file format to the SPPAS framework;
- export to, represented by an arrow from the SPPAS framework to the file format.

4.3 Supported file formats

SPPAS supports the following software with their file extensions:

- Praat: TextGrid, PitchTier, IntensityTier
- Elan: eaf
- Annotation Pro: ant, antx
- Phonedit: mrk
- Sclite: ctm, stm
- HTK: lab, mlf
- Subtitles: srt, sub
- Signaix: hz

- Spreadsheets, R,...: csv
- Audacity, notepads: txt

The followings can be imported:

- ANVIL: anvil
- Transcriber: trs
- Xtrans: tdf

The followings can be exported:

- Weka: arff, xrff

The support of external formats is regularly extended to new formats by the author on-demand from the users and distributed to the community in the SPPAS regular updates.

Data Analyses

5.1 Introduction

The analyses of annotated files includes the descriptive statistics, the filtering of the annotated data to get only the annotations you are interested in, see/edit the information about files, etc.

Like the other features of SPPAS, analyzing data can be performed in three different ways:

- the Application Programming Interface, in Python language;
- the Command-Line User Interface;
- the Graphical User Interface.

Among the features implemented in the API, a big majority are included in the GUI but just a few can be performed with the CLI. This chapter then describes only the page “Analyze” of the GUI.

5.2 The page Analyze of the GUI

The page “Analyze” of the GUI is divided into 2 main areas: a toolbar and a content to represent files.

5.2.1 Displayed files content

Each opened file is displayed in a panel in the content area. The next Figure shows how panels of 4 different files look like. Panels of the annotated files have a yellow background, panels of the audio files are blue and panels of unknown files are pinky-red.

Some actions can be performed individually on the panel of a file. A mouse click on the filename or on the arrow at left will show or hide the content of the file.

For transcription files, the icons at top-right of the panel allow to:

- edit the metadata of the file - notice that only the XRA file format allows to save the metadata;
- select the file (for example, it's used to paste tiers into);

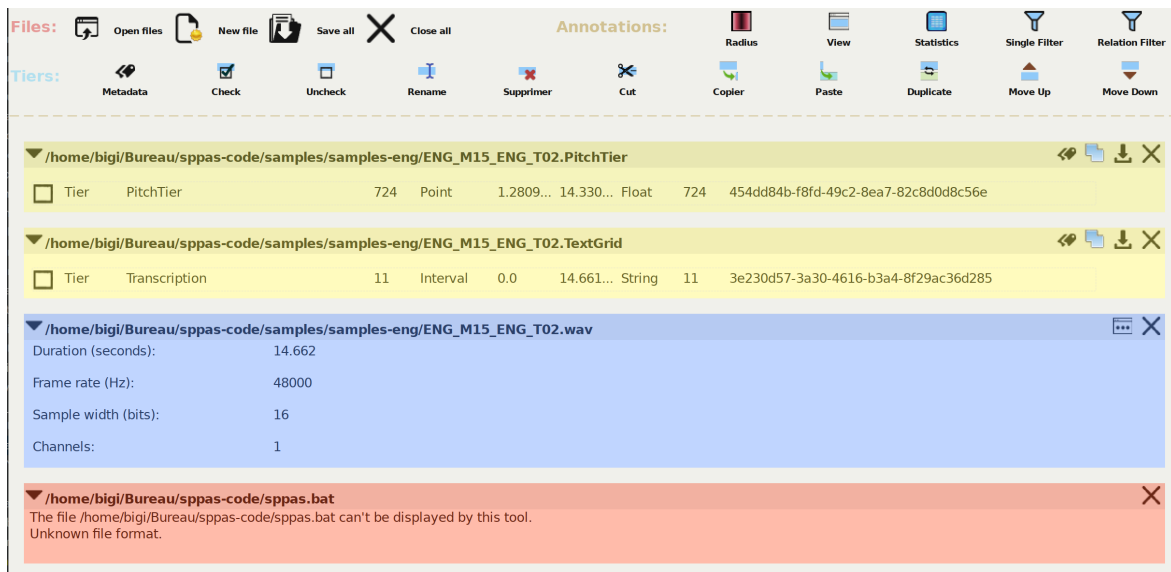


Figure 5.1: Analyze: open files

- save the file - only if it has changed;
- close the file and unlock it in the workspace.

For audio files, the icons at top-right of the panel allow to:

- view audio content and manage channels - available only for files less than a few minutes;
- close the file.

5.2.2 The toolbar

The toolbar of the page is made of 3 different parts:

1. “Files:” click on these buttons to perform an action on the checked filenames of the page “Files”;
2. “Tiers:” click on these buttons to perform an action on the checked tiers of the opened files;
3. “Annotations:” click on these buttons to analyze the annotations of the checked tiers of the opened files.

Files: Open files

Open, load and display a panel for each checked file of the current workspace. Some files could need a long time to be loaded (like TextGrid files with a lot of annotations), so a scrollbar should indicate the progression. As soon as a file is opened, it is locked and no other page can perform an action on it.

To open new files, check new files in the current workspace and click again on the “Open files” button. The panels of the newly opened files will be appended to the existing ones.

Files: New file

Click on this button to create a new annotated file. A dialog box will ask for a path/filename and for a file extension. The extension defines the file format; any of the supported file format can be used. The file will be created on disk when it will be saved for the first time.

Files: Save all

Save all files for which some changes were done, without confirmation.

Files: Close all

Close all the opened files. If some files were changed and not saved, a dialog will ask for confirmation.

Tiers: Metadata

Open a dialog to edit the metadata of the checked tiers. Notice that most of the file format don't allow to save metadata or it allows only some specific ones; only the XRA format allows to save any tier metadata.

Tiers: Check

A click on the "Check" button will open a dialog to enter a tier name and it will check all tiers matching it. The entry is a regular expression.

Tiers: Uncheck

A click on the "Uncheck" button will un-check all tiers.

Tiers: Rename

A mouse click on the "Rename" button will open a dialog to fix a name of a tier and then to rename all checked tiers. If a file has already a tier with the given name, an index number will be appended to the new name.

Tiers: Delete

A mouse click on the "Delete" button will delete all checked tiers. This process is irreversible. To recover a deleted tier, the only way is to close the file and to re-open it but all the un-saved changes are lost.

Tiers: Cut/Copy/Paste

The Cut/Copy/Paste buttons allow to use a clipboard to manage tiers. Tiers can then be copied from files to other ones. The files to paste in must be selected first with the "select" button of the individual panels.

Tiers: Duplicate

The duplicate button allows to copy/paste a tier into the same file. The name of the duplicated tier will be the same as the original one with an index number at the end.

Tiers: Move Up/Move Down

These buttons allow to move checked tiers into a file.

Annotations: Radius

The “Radius” button allows to fix a radius for all the localizations of the annotations of the checked tiers, i.e. the vagueness around the fixed point in time. Notice that only XRA file format allows to save it.

Read the following paper for details:

Brigitte Bigi, Tatsuya Watanabe, Laurent Prévot (2014). **Representing Multimodal Linguistics Annotated Data**, 9th International conference on Language Resources and Evaluation (LREC), Reykjavik (Iceland), pages 3386-3392. ISBN: 978-2-9517408-8-4.

Annotations: View

Click the button “View” to see all the annotations of the checked tiers in a table.

Annotations: Statistics

It allows to estimate the number of occurrences, the duration, etc. of the annotations of the checked tiers, and allows to save result in CSV (for Excel, OpenOffice, R, MatLab,...).

It offers a serie of sheets organized in a notebook. The first tab is displaying a summary of descriptive statistics of the set of given tiers. The other tabs are indicating one of the statistics over the given tiers. The followings are estimated:

- occurrences: the number of observations
- total durations: the sum of the durations
- mean durations: the arithmetic mean of the duration
- median durations: the median value of the distribution of durations
- std dev. durations: the standard deviation value of the distribution of durations

All of them can be estimated on a single annotation label or on a serie of them. The length of this context can be optionally changed while fixing the “N-gram” value (available from 1 to 5), just above the sheets.

Each displayed sheet can be saved as a CSV file, which is a useful file format to be read by R, Excel, OpenOffice, LibreOffice, and so... To do so, display the sheet you want to save and click on the button “Save sheet”, just below the sheets. If you plan to open this CSV file with Excel under Windows, it is recommended to change the encoding to UTF-16. For the other cases, UTF-8 is probably the most relevant.

The annotation durations are commonly estimated on the Midpoint value, without taking the radius into account; see (Bigi et al, 2012) for explanations about the Midpoint/Radius. Optionally, the duration can

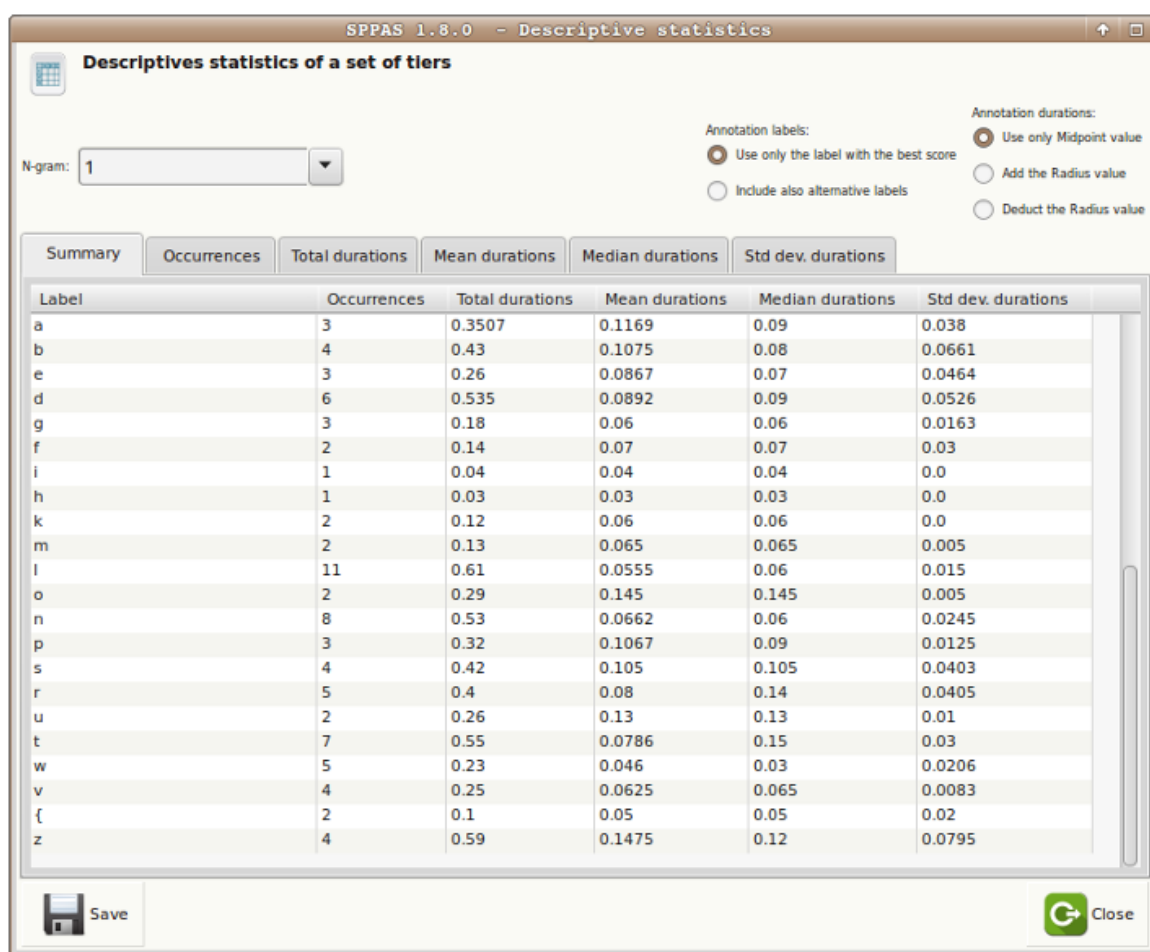


Figure 5.2: Descriptive statistics

either be estimated by taking the vagueness into account, then check “Add the radius value” button, or by ignoring the vagueness and estimating only on the central part of the annotation, then check “Deduct the radius value”.

For those who are estimating statistics on XRA files, you can either estimate stats only on the best label (the label with the higher score) or on all labels, i.e. the best label and all its alternatives (if any).

Annotations: Single filter

It allows to create a new tier with only selected annotations: define filters in order to create new tiers with only the annotations you are interested in!

Pattern selection is an important part to extract data of a corpus and is obviously and important part of any filtering system. Thus, if the label of an annotation is a string, the following filters are proposed in DataFilter:

- exact match: an annotation is selected if its label strictly corresponds to the given pattern;
- contains: an annotation is selected if its label contains the given pattern;
- starts with: an annotation is selected if its label starts with the given pattern;
- ends with: an annotation is selected if its label ends with the given pattern.

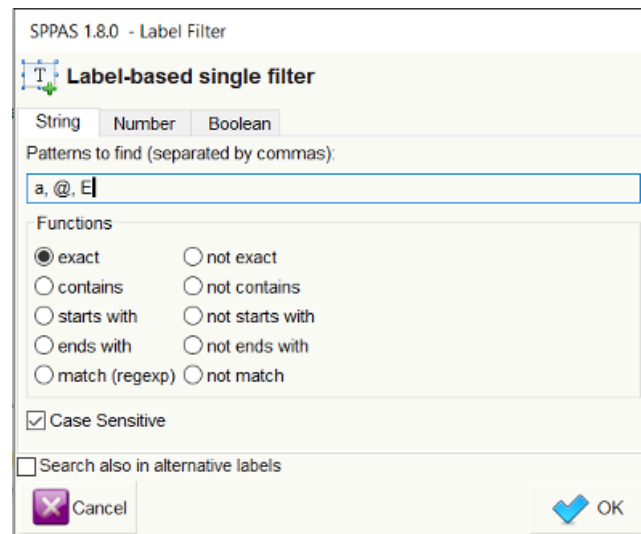


Figure 5.3: Frame to create a filter on annotation label tags: filter annotations that exactly match either a, @ or E

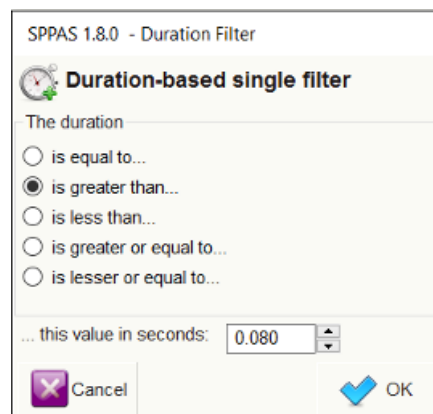


Figure 5.4: Frame to create a filter on annotation durations: filter annotations that are during more that 80 ms

All these matches can be reversed to represent respectively: does not exactly match, does not contain, does not start with or does not end with. Moreover, this pattern matching can be case sensitive or not.

For complex search, a selection based on regular expressions is available for advanced users.

A multiple pattern selection can be expressed in both ways:

- enter multiple patterns at the same time (separated by commas) to mention the system to retrieve either one pattern or the other, etc.
- enter one pattern at a time and choose the appropriate button: “Apply All” or “Apply any”.

Another important feature for a filtering system is the possibility to retrieve annotated data of a certain duration, and in a certain range of time in the timeline.

Search can also starts and/or ends at specific time values in a tier.

In SPPAS 3.7, a new filter is added: it can select annotation depending on their number of labels. For example, the automatic annotation “Normalization” creates a tier “Tokens” in which each annotation contains a list of

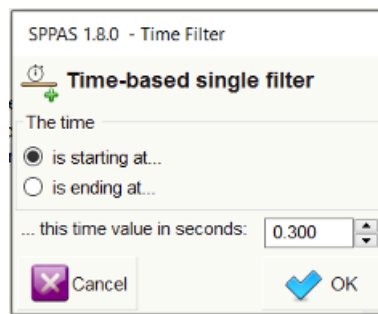


Figure 5.5: Frame to create a filter on annotation time values: filter annotations that are starting after the 5th minute

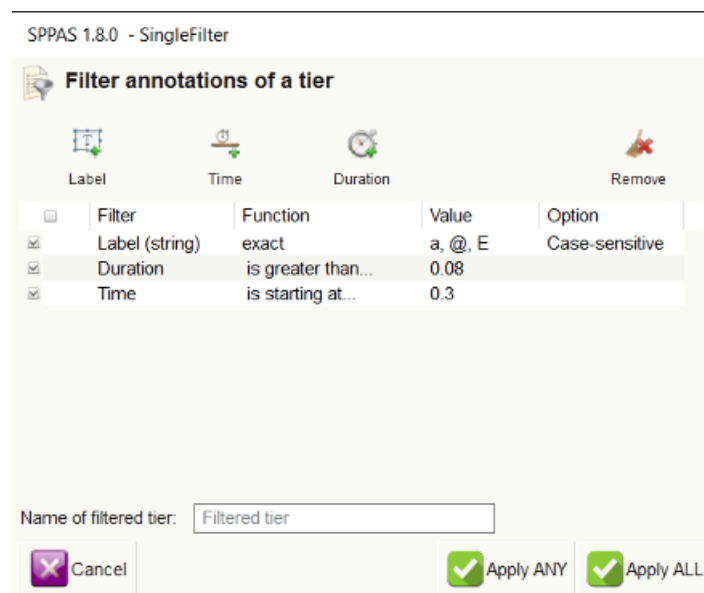


Figure 5.6: DataFilter: SingleFilter frame

labels - one per token; then it is possible to get the annotations with more than 3 tokens, with only one token, etc.

All the given filters are summarized in the “SingleFilter” dialog. To complete the filtering process, it must be clicked on one of the apply buttons and the new resulting tiers are added in the annotation file(s).

In the given example:

- click on “Apply All” to get either a, @ or E vowels during more than 80ms, after the 5th minute.
- click on “Apply Any” to get a, @ or E vowels, and all annotations during more than 80 ms, and all annotations after the 5th minute.

Read the following publications for details:

Brigitte Bigi (2019). Filtering multi-levels annotated data. In 9th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 13-14, Poznań, Poland.

Brigitte Bigi, Jorane Saubesty (2015). Searching and retrieving multi-levels annotated data, Proceedings of Gesture and Speech in Interaction, Nantes (France).

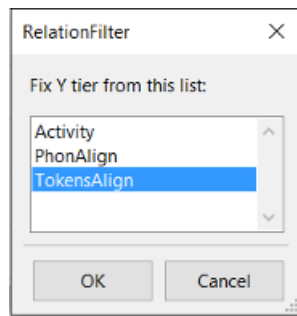


Figure 5.7: Fix time-relation tier name

Annotations: Relation filter

Regarding the searching problem, linguists are typically interested in locating patterns on specific tiers, with the possibility to relate different annotations a tier from another. The proposed system offers a powerful way to request/extract data, with the help of Allen’s interval algebra.

In 1983 James F. Allen published a paper in which he proposed 13 basic relations between time intervals that are distinct, exhaustive, and qualitative:

- distinct because no pair of definite intervals can be related by more than one of the relationships;
- exhaustive because any pair of definite intervals are described by one of the relations;
- qualitative (rather than quantitative) because no numeric time spans are considered.

These relations and the operations on them form Allen’s interval algebra. These relations were extended to Interval-Tiers as Point-Tiers to be used to find/select/filter annotations of any kind of time-aligned tiers.


For the sake of simplicity, only the 13 relations of the Allen’s algebra are available in the GUI. But actually, we implemented the 25 relations proposed Pujari and al. (1999) in the INDU model. This model is fixing constraints on INtervals (with Allen’s relations) and on DUration (duration are equals, one is less/greater than the other). Such relations are available while requesting with Python.

At a first stage, the user must select the tiers to be filtered and click on “RelationFilter”. The second stage is to select the tier that will be used for time-relations.

The next step consists in checking the Allen’s relations that will be applied. The last stage is to fix the name of the resulting tier. The above screenshots illustrates how to select the first phoneme of each token, except for tokens that are containing only one phoneme (in this later case, the “equal” relation should be checked).

To complete the filtering process, it must be clicked on the “Apply” button and the new resulting tiers are added in the annotation file(s).

SPPAS 1.8.0 - RelationFilter

 **Filter a tier X, depending on time-relations with a tier Y**

Tier X: PhonAlign

Tier Y: TokensAlign Fix name

<input checked="" type="checkbox"/> Equals		X ---- Y ----	Non efficient	Non efficient	X Y
<input type="checkbox"/> Before	Max delay in seconds: 3.000	X ---- Y ----	X ---- Y	X Y ----	X Y
<input type="checkbox"/> After	Max delay in seconds: 3.000	X ---- Y ----	X Y ----	X ---- Y	X Y
<input type="checkbox"/> Meets		X ---- Y ----	Non efficient	Non efficient	Non efficient
<input type="checkbox"/> Met by		X ---- Y ----	Non efficient	Non efficient	Non efficient
<input type="checkbox"/> Overlaps	Min overlap in seconds: 0.001	X ---- Y ----	Non efficient	Non efficient	Non efficient
<input type="checkbox"/> Overlapped by	Min overlap in seconds: 0.001	X ---- Y ----	Non efficient	Non efficient	Non efficient
<input checked="" type="checkbox"/> Starts		X --- Y ----	Non efficient	Non efficient	Non efficient
<input checked="" type="checkbox"/> Started by		X ---- Y ---	Non efficient	Non efficient	Non efficient

☐ Replace label of X by the relation name.

Name of filtered tier:



 Cancel  OK

Figure 5.8: DataFilter: RelationFilter frame

Scripting with Python and SPPAS

6.1 Introduction

Since version 1.8.7, SPPAS implements an Application Programming Interface (API), named *anndata*, to deal with annotated files. Previously, SPPAS was based on another API named *annotationdata* which is still distributed in the package but no longer updated or maintained.

anndata API is a free and open source Python library to access and search data from annotated data of any of the supported formats (xra, TextGrid, eaf...). It can either be used with the Programming Language Python 2.7 or Python 3.4+. This API is PEP8 and PEP257 compliant and the internationalization of the messages is implemented (English and French are available in the “po” directory).

In this chapter, it is assumed that a version of Python is installed and configured. It is also assumed that the Python IDLE is ready-to-use. For more details about Python, see:

The Python Website: <http://www.python.org>

This chapter firstly introduces basic programming concepts, then it gradually introduces how to write scripts with Python. Those who are familiar with programming in Python can directly go to the last section related to the description of the *anndata* API and how to use it in Python scripts.

This API can convert file formats like Elan’s EAF, Praat’s TextGrid and others into a *sppasTranscription* object and convert this object into any of these formats. This object allows unified access to linguistic data from a wide range sources.

This chapter includes exercises. The solution scripts are included in the package directory **documentation**, folder **scripting_solutions**.

6.2 A gentle introduction to programming



Figure 6.1: The Python IDLE logo

6.2.1 Introduction

This section includes examples in Python programming language. You may want to try out some of the examples that come with the description. In order to do this, execute the Python IDLE - available in the Application-Menu of your operating system, and write the examples after the prompt “>>>”.

To get information about the IDLE, get access to [the IDLE documentation](#)

Writing any program consists of writing statements so using a programming language. A *statement* is often known as a line of code that can be one of:

- comment,
- documentation,
- assignment,
- conditions,
- iterations, etc.

Lines of code are grouped in *blocks*. Depending on the programming language, blocks delimited by brackets, braces or by the indentation.

Each language has its own syntax to write these lines and the user has to follow strictly this syntax for the program to be able to interpret the program. However, the amount of freedom the user has to use capital letters, whitespace and so on is very high. Recommendations for Python language are available in the [PEP8 - Style Guide for Python Code](#).

6.2.2 Variables: Assignment and Typing

A *variable* is a name to give to a piece of memory with some information inside. Assignment is then the action of setting a variable to a value. The equal sign (=) is used to assign values to variables.

```
>>>a = 1
>>>b = 1.0
>>>c = "c"
>>>hello = "Hello world!"
>>>vrai = True
```

In the previous example, `a`, `b`, `c`, `hello` and `vrai` are variables, `a = 1` is a declaration.

Assignments to variables with Python language can be performed with the following operators:

```
>>> a = 10    # simple assignment operator
>>> a += 2     # add and assignment operator, so a is 12
>>> a -= 7     # minus and assignment, so a is 5
```

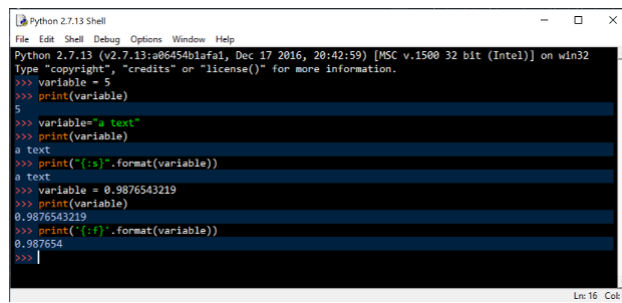


Figure 6.2: Variable declarations and print in the Python IDLE

```
>>> a *= 20 # multiply and assignment, so a is 100
>>> a /= 10 # divide and assignment, so a is 10
>>> a      # verify the value of a...
10
```

6.2.3 Basic Operators

Basic operators are used to manipulate variables. The following is the list of operators that can be used with Python, i.e. equal (assignment), plus, minus, multiply, divide:

```
>>> a = 10
>>> b = 20 # assignment
>>> a + b   # addition
>>> a - b   # subtraction
>>> a * b   # multiplication
>>> a / b   # division
```

6.2.4 Data types

The variables are of a data-type. For example, the declarations `a=1` and `a=1.0` are respectively assigning an integer and a real number. In Python, the command `type` allows to get the type of a variable, like in the following:

```
>>> type(a)
<type 'int'>
>>> type(b)
<type 'float'>
>>> type(c)
<type 'str'>
>>> type(cc)
<type 'unicode'>
>>> type(vrai)
<type 'bool'>
```

Here is a list of some fundamental data types, and their characteristics:

- *str* String of characters
- *unicode* Unicode string of characters
- *int* Integer in range from -2147483648 to 2147483647
- *bool* Boolean value, can take value `True` (=1) or `False` (=0)
- *float* Floating point number (max 15 digits)

Python is assigning data types dynamically. As a consequence, the result of the sum between an `int` and a `float` is a `float`. The next examples illustrate that the type of the variables have to be carefully managed.

```
>>> a = 10
>>> a += 0.
>>> a
10.0
>>> a += True
>>> a
11.0
>>> a += "a"
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: unsupported operand type(s) for +=: 'float' and 'str'
>>> a = "a"
>>> a *= 5
>>> a
'aaaaa'
```

The type of a variable can be explicitly changed. This is called a “cast”:

```
>>> a = 10
>>> b = 2
>>> a/b
5
>>> float(a) / float(b)
5.0
>>> a = 1
>>> b = 1
>>> str(a) + str(b)
'11'
```

Complex data types are often used to store variables sharing the same properties like a list of numbers, and so on. Common types in languages are lists/arrays and dictionaries. The following is the assignment of a list with name `fruits`, then the assignment of a sub-part of the list to the `to_buy` list:

```
>>> fruits = ['apples', 'tomatoes', 'peers', 'bananas', 'lemons']
>>> to_buy = fruits[1:3]
>>> to_buy
['tomatoes', 'peers']
```

6.2.5 Conditions

Conditions aim to test whether a statement is True or False. The statement of the condition can include a variable, or be a variable and is written with operators. The following shows examples of conditions/comparisons in Python. Notice that the comparison of variables of a different data-type is possible (but not recommended!).

```
>>> var = 100
>>> if var == 100:
...     print("Value of expression is 100.")
...
Value of expression is 100.

>>> if var == "100":
...     print("This message won't be printed out.")
...

```

Conditions can be expressed in a more complex way like:

```
>>> if a == b:
...     print('a and b are equals')
... elif a > b:
...     print('a is greater than b')
... else:
...     print('b is greater than a')

```

The simple operators for comparisons are summarized in the next examples:

```
>>> a == b    # check if equals
>>> a != b    # check if different
>>> a > b     # check if a is greater than b
>>> a >= b    # check if a is greater or equal to b
>>> a < b     # check if a is lesser than b
>>> a <= b    # check if a is lesser or equal to b

```

It is also possible to use the following operators:

- **and**: called “Logical AND operator”. If both the operands are true then the condition becomes true.
- **or**: called “Logical OR operator”. If any of the two operands are non zero then the condition becomes true.
- **not**: called “Logical NOT operator”. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.
- **in**: evaluates to true if it finds a variable in the specified sequence and false otherwise.

```
>>> if a == "apples" and b == "peers":
...     print("You need to buy fruits.")

```



```
>>> if a == "apples" or b == "apples":
...     print("You already have bought apples.")

>>> if "tomatoas" not in to_buy:
...     print("You don't have to buy tomatoes.")
```

6.2.6 Loops

The `for` loop statement iterates over the items of any sequence. The next Python lines of code print items of a list on the screen:

```
>>> to_buy = ['fruits', 'viande', 'poisson', 'oeufs']
>>> for item in to_buy:
...     print(item)
...
fruits
viande
poisson
oeufs
```

A `while` loop statement repeatedly executes a target statement as long as a given condition returns `True`. The following example prints exactly the same result as the previous one:

```
>>> to_buy = ['fruits', 'viande', 'poisson', 'oeufs']
>>> i = 0
>>> while i < len(to_buy):
...     print(to_buy[i])
...     i += 1
...
fruits
viande
poisson
oeufs
```

6.2.7 Dictionaries

A dictionary is a very useful data type. It consists of pairs of keys and their corresponding values.

```
>>> fruits = dict()
>>> fruits['apples'] = 3
>>> fruits['peers'] = 5
>>> fruits['tomatoas'] = 1
```

`fruits['apples']` is a way to get the value - i.e. 3, of the apple key. However, an error is sent if the key is unknown, like `fruits[bananas]`. Alternatively, the `get` function can be used, like `fruits.get("bananas", 0)` that returns 0 instead of an error.

The next example is showing how use a simple dictionary:

```
>>> for key in fruits:
...     value = fruits.get(key, 0)
...     if value < 3:
...         print("You have to buy new {:s}.".format(key))
...
You have to buy new tomatoes.
```

To learn more about data structures and how to manage them, get access to [the Python documentation](#)

6.3 Scripting with Python

This section describes how to create simple Python lines of code in separated files commonly called *scripts*, and run them. Some practical exercises, appropriate to the content of each action, are proposed and test exercises are suggested at the end of the section.

To practice, you have first to create a new folder in your computer - on your Desktop for example; with name “pythonscripts” for example, and to execute the python IDLE.

For an advanced use of Python, the installation of a dedicated IDE is very useful. SPPAS is developed with PyCharm: See [the PyCharm Help webpage](#)

6.3.1 Comments and documentation

Comments are not required by the program to work. But comments are necessary! Comments are expected to be appropriate, useful, relevant, adequate and always reasonable.

```
# This script is doing this and that.
# It is under the terms of a license.
# and I can continue to write what I want after the # symbol
# except that it's not the right way to tell the story of my life
```

The documentation of a program complements the comments. Both are not sharing the same goal: comments are used in all kind of programs but documentation is appended to comments for the biggest programs and/or projects. Documentation is automatically extracted and formatted thanks to dedicated tools. Documentation is required for sharing the program. See the [Docstring Conventions](#) for details. Documentation must follow a convention like for example the markup language [reST - reStructured Text](#). Both conventions are used into SPPAS API, programs and scripts.

6.3.2 Getting started with scripting in Python

In the IDLE, create a new empty file either by clicking on “File” menu, then “New File”, or with the shortcut “CTRL”+N.

Copy the following line of code in this newly created file:

```
1 print("Hello world!")
```

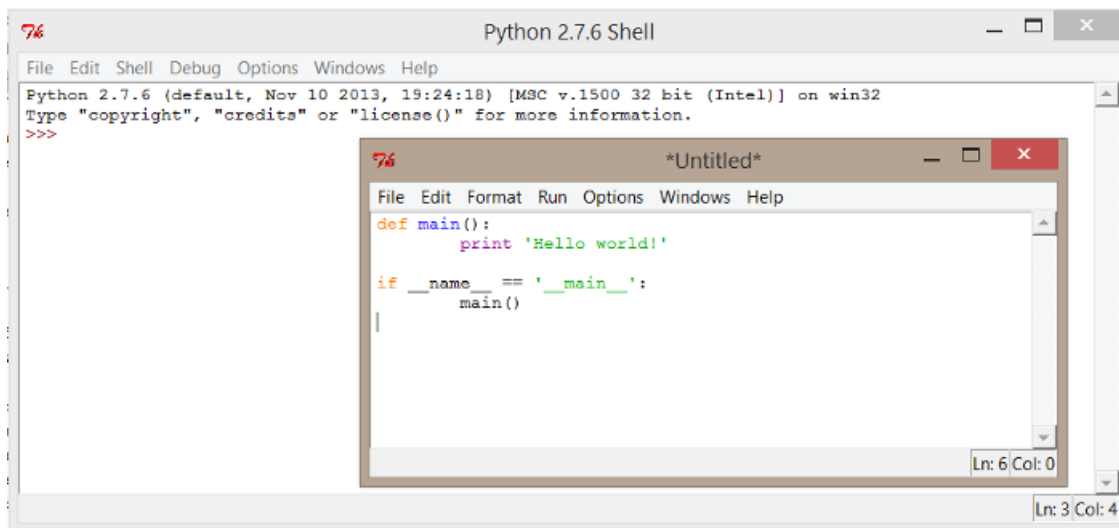


Figure 6.3: Hello world! in a Python script

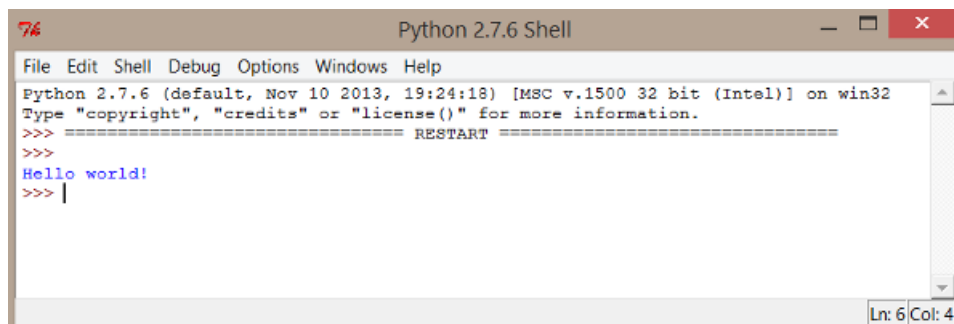


Figure 6.4: Output of the first script

Then, save the file in the “pythonscripts” folder. By convention, Python source files end with a `.py` extension, and so the name `01_helloworld.py` could be fine.

To execute the program, you can do one of:

- with the mouse: Click on the Menu “Run”, then “Run module”
- with the keyboard: Press F5

The expected output is as follow:

A better practice while writing scripts is to describe by who, what and why this script was done. A nifty trick is to create a skeleton for any future script that will be written. Such ready-to-use script is available in the SPPAS package with the name `skeleton.py`.

6.3.3 Blocks

Blocks in Python are created from the indentation. Tab and spaces can be used but using spaces is recommended.

```
>>> if a == 3:
...     # this is a block using 4 spaces for indentation
...     print("a is 3")
```

6.3.4 Functions

Simple function

A function does something: it starts with its definition then is followed by its lines of code in a block.

Here is an example of function:

```
27 def print_vowels():
28     """ Print the list of French vowels on the screen. """
29
30     vowels = ['a', 'e', 'E', 'i', 'o', 'u', 'y', '@', '2', '9', 'a~', 'o~', 'U~']
31     print("List of French vowels:")
32     for v in vowels:
33         print(v)
```

What the `print_vowels()` function is doing? This function declares a list with name `vowels`. Each item of the list is a string representing a vowel in French encoded in X-SAMPA. Of course, this list can be overridden with any other set of strings. The next line prints a message. Then, a loop prints each item of the list.

At this stage, if a script with this function is executed, it will do... nothing! Actually, the function is created, but it must be invoked in the main function to be interpreted by Python. The `main` is as follow:

```
34 if __name__ == '__main__':
35     print_vowels()
```

Practice: create a copy of the file `skeleton.py`, then make a function to print “Hello World!”. (solution: `ex01_hello_world.py`).

Practice: Create a function to print plosives and call it in the main function (solution: `ex02_functions.py`).

One can also create a function to print glides, another one to print affricates, and so on. Hum... this sounds a little bit fastidious!

Function with parameters

Rather than writing the same lines of code with only a minor difference over and over, we can declare *parameters* to the function to *make it more generic*. Notice that the number of parameters of a function is not limited!

In the example, we can replace the `print_vowels()` function and the `print_plosives()` function by a single function `print_list(mylist)` where `mylist` can be any list containing strings or characters. If the list contains other typed-variables like numerical values, they must be converted to string to be printed out. This can result in the following function:

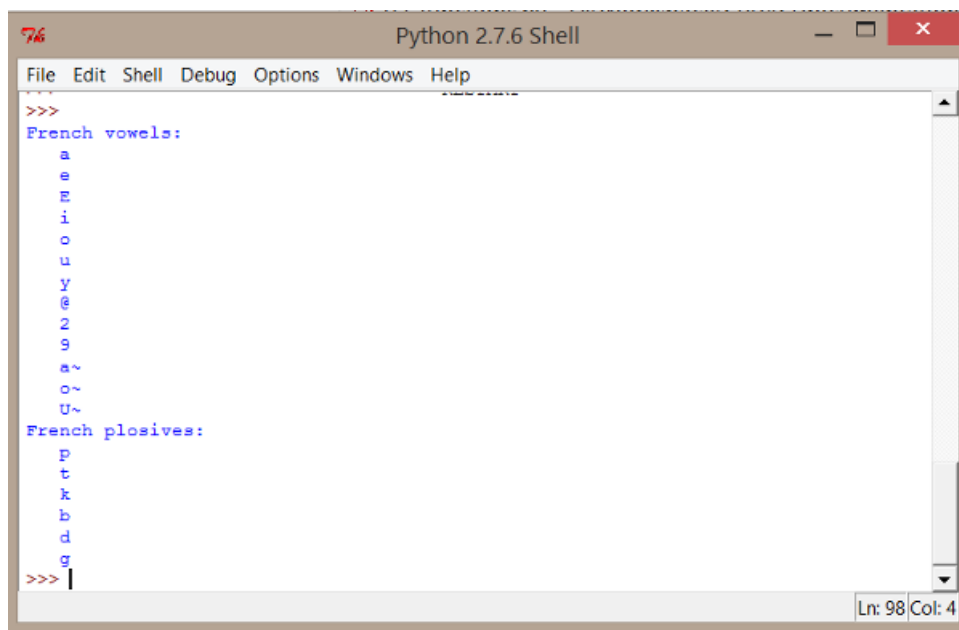


Figure 6.5: Output of the second script

```

27 def print_list(mylist, message="-"):
28     """ Print a list on the screen.
29
30     :param mylist: (list) the list to print
31     :param message: (string) an optional message to print before each element
32
33     """
34     for item in mylist:
35         print("{:s} {:s}".format(message, item))

```

Function return values

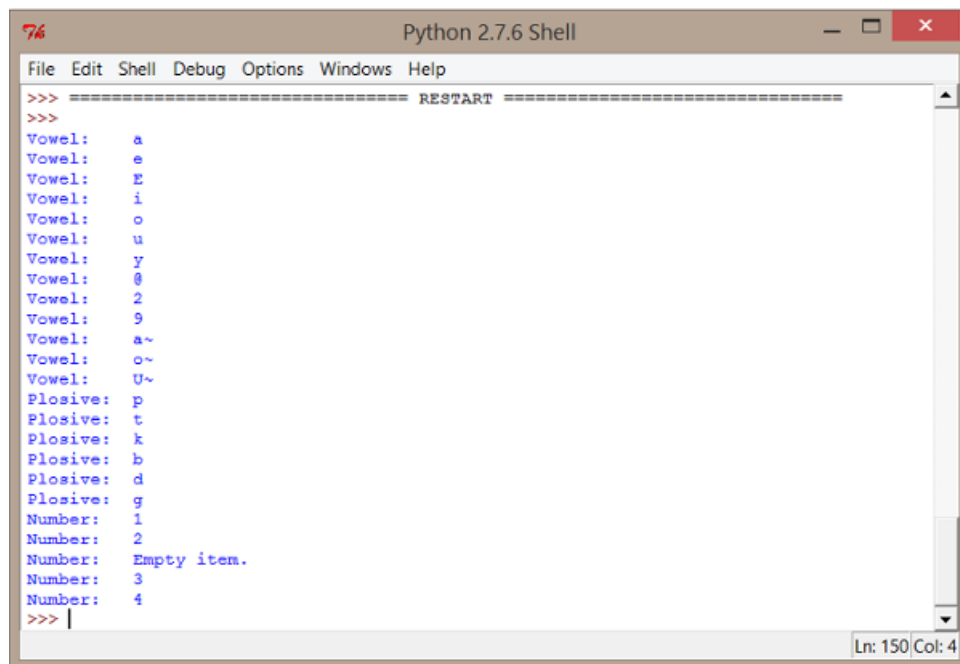
Functions are used to do a specific job and the result of the function can be captured by the program. In the following example, the function would return a boolean value, i.e. True if the given string has no character.

```

27 def is_empty(mystr):
28     """ Return True if mystr is empty. """
29
30     return len(mystr.strip()) == 0

```

Practice: Add this function in a new script and try to print various lists (solution: ex03_functions.py)



```

Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
>>> ===== RESTART =====
>>>
Vowel: a
Vowel: e
Vowel: E
Vowel: i
Vowel: o
Vowel: u
Vowel: y
Vowel: @
Vowel: 2
Vowel: 9
Vowel: a~
Vowel: o~
Vowel: U~
Plosive: p
Plosive: t
Plosive: k
Plosive: b
Plosive: d
Plosive: g
Number: 1
Number: 2
Number: Empty item.
Number: 3
Number: 4
>>> |
Ln: 150 Col: 4

```

Figure 6.6: Expected output of the 3rd script

6.3.5 Reading/Writing files

Reading data from a file

Now, we'll try to get data from a file. Create a new empty file with the following lines - and add as many lines as you want; then, save it with the name "phonemes.csv" by using UTF-8 encoding:

```

occlusives ; b ; b
occlusives ; d ; d
fricatives ; f ; f
liquids ; l ; l
nasals ; m ; m
nasals ; n ; n
occlusives ; p ; p
glides ; w ; w
vowels ; a ; a
vowels ; e ; e

```

The following statements are typical statements used to read the content of a file. The first parameter of the `open` function is the name of the file, including the path (relative or absolute); and the second argument is the opening mode ('r' is the default value, used for reading).

Practice: Add these lines of code in a new script and try it (solution: `ex04_reading_simple.py`)

```

21 fp = open("phonemes.csv", 'r')
22 for line in fp:
23     # do something with the line stored in variable l
24     print(line.strip())
25 f.close()

```

The following is a solution with the ability to deal with various file encodings, thanks to the `codecs` library:

```
21 def read_file(filename):
22     """ Get the content of file.
23
24     :param filename: (string) Name of the file to read, including path.
25     :returns: List of lines
26
27     """
28     with codecs.open(filename, 'r', encoding="utf8") as fp:
29         return fp.readlines()
```

In the previous code, the `codecs.open` functions got 3 parameters: the name of the file, the mode to open, and the encoding. The `readlines()` function gets each line of the file and store it into a list.

Practice: Write a script to print the content of a file (solution: `ex05_reading_file.py`)

Notice that Python `os` module provides useful methods to perform file-processing operations, such as renaming and deleting. See Python documentation for details: <https://docs.python.org/2/>

Writing data to a file

Writing a file requires to open it in a writing mode:

- 'w' is the mode to write data; it will erase any existing file;
- 'a' is the mode to append data in an existing file.

A file can be opened in an encoding and saved in another one. This could be useful to write a script to convert the encoding of a set of files. The following could help to create such script:

```
10 # getting all files of a given folder:
11 path = 'C:\Users\Me\data'
12 dirs = os.listdir( path )
13
14
15 # Converting the encoding of a file:
16 file_stream = codecs.open(file_location, 'r', file_encoding)
17 file_output = codecs.open(file_location+'utf8', 'w', 'utf-8')
18
19 for line in file_stream:
20     file_output.write(line)
```

6.3.6 Python tutorials

Here is a list of web sites with tutorials, from the easiest to the most complete:

1. [Learn Python, by DataCamp](#)
2. [Tutorial Points](#)
3. [The Python documentation](#)

6.3.7 Exercises to practice

Exercise 1: How many vowels are in a list of phonemes? (solution: `ex06_list.py`)

Exercise 2: Write a X-SAMPA to IPA converter. (solution: `ex07_dict.py`)

Exercise 3: Compare 2 sets of data using NLP techniques (Zipf law, Tf.Idf) (solution: `ex08_counter.py`)

6.4 `anndata`, an API to manage annotated data

6.4.1 Overview

We are now going to write Python scripts using the `anndata` API included in SPPAS. This API is useful to read/write and manipulate files annotated from various annotation tools like SPPAS, Praat or Elan.

First of all, it is important to understand the data structure included into the API to be able to use it efficiently.

6.4.2 Why developing a new API?

In the Linguistics field, multimodal annotations contain information ranging from general linguistic to domain specific information. Some are annotated with automatic tools, and some are manually annotated. In annotation tools, annotated data are mainly represented in the form of “tiers” or “tracks” of annotations. Tiers are mostly series of intervals defined by:

- a time point to represent the beginning of the interval;
- a time point to represent the end of the interval;
- a label to represent the annotation itself.

Of course, depending on the annotation tool, the internal data representation and the file formats are different. In Praat, tiers can be represented by a time point and a label (such tiers are respectively named `PointTiers` and `IntervalTiers`). `IntervalTiers` are made of a succession of consecutive intervals (labelled or un-labelled). In Elan, points are not supported; and unlabelled intervals are not represented nor saved.

The `anndata` API was designed to be able to manipulate all data in the same way, regardless of the file type. It supports to merge data and annotations from a wide range of heterogeneous data sources.

6.4.3 The API class diagram

After opening/loading a file, its content is stored in a `sppasTranscription` object. A `sppasTranscription` has a name, and a list of `sppasTier` objects. Tiers can't share the same name, the list of tiers can be empty, and a hierarchy between tiers can be defined. Actually, subdivision relations can be established between tiers. For example, a tier with phonemes is a subdivision reference for syllables, or for tokens; and tokens are a subdivision reference for the orthographic transcription in IPUs. Such subdivisions can be of two categories: alignment or association.

A `sppasTier` object has a name, and a list of `sppasAnnotation` objects. It can also be associated to a controlled vocabulary, or a media.

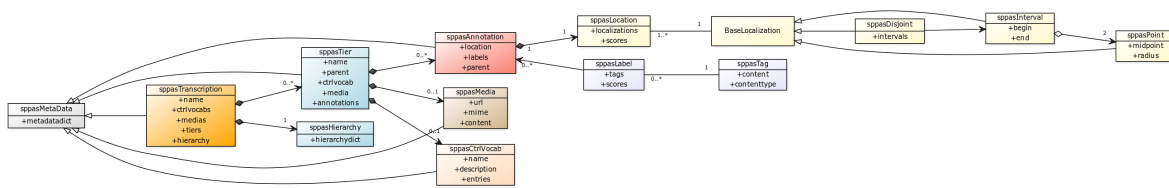


Figure 6.7: API class diagram

All these objects contain a set of meta-data.

An annotation is made of 2 objects:

- a `sppasLocation` object,
- a list of `sppasLabel` objects.

A `sppasLabel` object is representing the “content” of the annotation. It is a list of `sppasTag` each one associated to a score.

A `sppasLocation` is representing where this annotation occurs in the media. Then, a `sppasLocation` is made of a list of localization each one associated with a score. A localization is one of:

- a `sppasPoint` object; or
- a `sppasInterval` object, which is made of 2 `sppasPoint` objects; or
- a `sppasDisjoint` object which is a list of `sppasInterval`.

Label representation

Each annotation holds a series of 0..N labels, mainly represented in the form of a string, freely written by the annotator or selected from a list of categories.

Location representation

In the *anndata* API, a `sppasPoint` is considered as an *imprecise value*. It is possible to characterize a point in a space immediately allowing its vagueness by using:

- a midpoint value (center) of the point;
- a radius value.

Example

The screenshot below shows an example of multimodal annotated data, imported from 3 different annotation tools. Each `sppasPoint` is represented by a vertical dark-blue line with a gradient color to refer to the radius value.

In the screenshot the following radius values were assigned:

- 0ms for prosody,
- 5ms for phonetics, discourse and syntax
- 40ms for gestures.

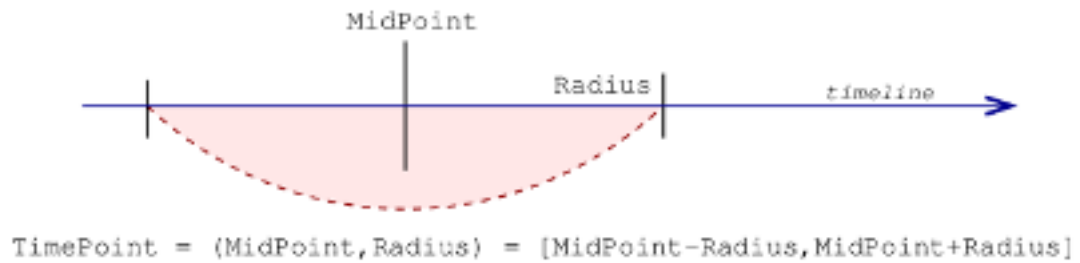


Figure 6.8: Representation of a sppasPoint

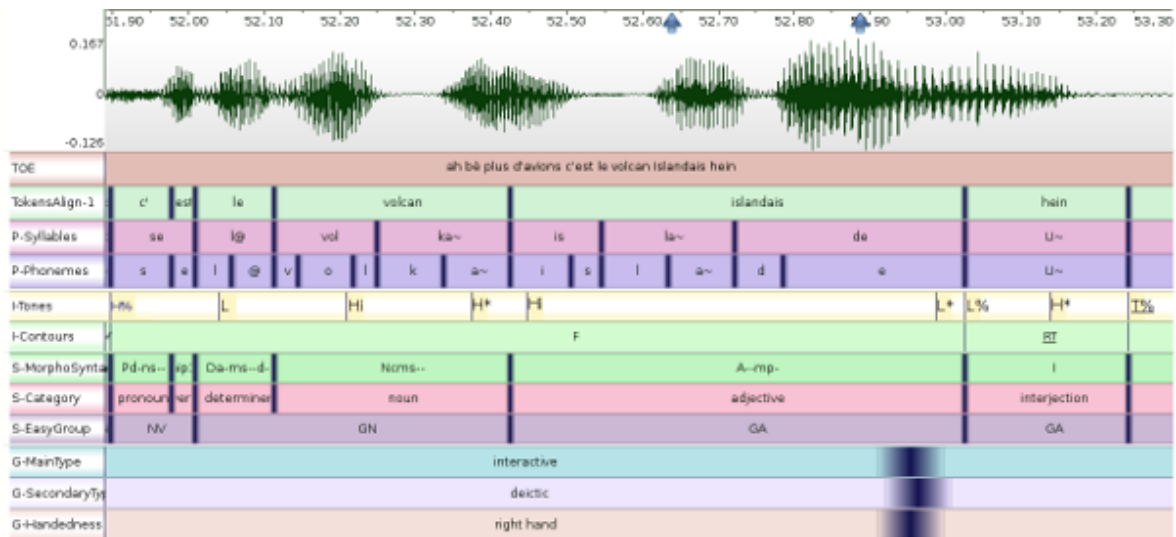


Figure 6.9: Example of multimodal data

6.5 Creating scripts with anndata

6.5.1 Preparing the data

To practice, you have first to create a new folder in your computer - on your Desktop for example; with name “sppasscripts” for example, and to execute the python IDLE.

Open a File Explorer window and go to the SPPAS folder location. Then, copy the `sppas` directory into the newly created “sppasscripts” folder. Then, go to the solution directory and copy/paste the files `skeleton-sppas.py` and `F_F_B003-P9-merge.TextGrid` into your “sppasscripts” folder. Then, open the skeleton script with the python IDLE and execute it. It will do... nothing! But now, you are ready to do something with the API of SPPAS!

When using the API, if something forbidden is attempted, the object will raise an Exception. It means that the program will stop except if the script “raises” the exception.

6.5.2 Read/Write annotated files

We are being to Open/Read an annotated file of any format (XRA, TextGrid, Elan, ...) and store it into a `sppasTranscription` object instance. Then, it will be saved into another file.

```
1  # Create a parser object then parse the input file.
2  parser = sppasRW(input_filename)
3  trs = parser.read()
4
5  # Save the sppasTranscription object into a file.
6  parser.set_filename(output_filename)
7  parser.write(trs)
```

Only these two lines of code are required to convert a file from a format to another one! The appropriate parsing system is extracted from the extension of file name.

To get the list of accepted extensions that the API can read, just use `aio.extensions_in`. The list of accepted extensions that the API can write is given by `aio.extensions_out`.

Practice: Write a script to convert a TextGrid file into CSV (solution: `ex10_read_write.py`)

6.5.3 Manipulating a `sppasTranscription` object

The most useful functions to manage tiers of a `sppasTranscription` object are:

- `create_tier()` to create an empty tier and to append it,
- `append(tier)` to add a tier into the `sppasTranscription`,
- `pop(index)` to remove a tier of the `sppasTranscription`,
- `find(name, case_sensitive=True)` to find a tier from its name.

```
15  for tier in trs:
16      # do something with the tier:
17      print(tier.get_name())
18  phons_tier = trs.find("PhonAlign")
```

Practice: Write a script to select a set of tiers of a file and save them into a new file (solution: `ex11_transcription.py`).

6.5.4 Manipulating a `sppasTier` object

A tier is made of a name, a list of annotations, and optionally a controlled vocabulary and a media. To get the name of a tier, or to fix a new name, the easier way is to use `tier.get_name()`. The following block of code allows to get a tier and change its name.

```
43 # Get the first tier, with index=0
44 tier = trs[0]
45 print(tier.get_name())
46 tier.set_name("NewName")
47 print(tier.get_name())
```

The most useful functions to manage annotations of a `sppasTier` object are:

- `create_annotation(location, labels)` to create and add a new annotation
- `append(annotation)` to add a new annotation at the end of the list
- `add(annotation)` to add a new annotation
- `pop(index)` to delete the annotation of a given index
- `remove(begin, end)` to remove annotations of a given localization range
- `is_disjoint()`, `is_interval()`, `is_point()` to know the type of location
- `is_string()`, `is_int()`, `is_float()`, `is_bool()` to know the type of labels
- `find(begin, end)` to get annotations in a given localization range
- `get_first_point()`, `get_last_point()` to get respectively the point with the lowest or highest localization
- `set_radius(radius)` to fix the same vagueness value to each localization point

Practice: Write a script to open an annotated file and print information about tiers (solution: `ex12_tiers_info.py`)

6.5.5 Manipulating a `sppasAnnotation` object

An annotation is a container for a location and optionally a list of labels. It can be used to manage the labels and tags with the following methods:

- `is_labelled()` returns `True` if at least a `sppasTag` exists and is not `None`
- `append_label(label)` to add a label at the end of the list of labels
- `get_labels_best_tag()` returns a list with the best tag of each label
- `add_tag(tag, score, label_index)` to add a tag into a label
- `remove_tag(tag, label_index)` to remove a tag of a label
- `serialize_labels()` to get a string representing the sequence of labels

An annotation object can also be copied with the method `copy()`. The location, the labels and the metadata are all copied; and the 'id' of the returned annotation is then the same. It is expected that each annotation of a tier as its own 'id', but the API doesn't check this.

Practice: Write a script to print information about annotations of a tier (solution: `ex13_tiers_info.py`)

6.5.6 Search in annotations: Filters

Overview

This section focuses on the problem of *searching and retrieving* data from annotated corpora.

The filter implementation can only be used together with the `sppasTier()` class. The idea is that each `sppasTier()` can contain a set of filters, that each reduce the full list of annotations to a subset.

SPPAS filtering system proposes 2 main axis to filter such data:

- with a boolean function based either on the content, the duration or on the time of annotations,
- with a relation function between annotation locations of 2 tiers.

A set of filters can be created and combined to get the expected result. To be able to apply filters to a tier, some data must be loaded first. First, a new `sppasTranscription()` has to be created when loading a file. Then, the tier(s) to apply filters on must be fixed. Finally, if the input file was NOT an XRA, it is widely recommended to fix a radius value before using a relation filter.

```
f = sppasFilter(tier)
```

When a filter is applied, it returns an instance of `sppasAnnSet` which is the set of annotations matching with the request. It also contains a 'value' which is the list of functions that are truly matching for each annotation. Finally, `sppasAnnSet` objects can be combined with the operators '!' and '&', and expected to a `sppasTier` instance.

Filter on the tag content

The following matching names are proposed to select annotations:

- 'exact': means that a tag is valid if it strictly corresponds to the expected pattern;
- 'contains' means that a tag is valid if it contains the expected pattern;
- 'startswith' means that a tag is valid if it starts with the expected pattern;
- 'endswith' means that a tag is valid if it ends with the expected pattern.
- 'regex' to define regular expressions.

All these matches can be reversed, to represent does not exactly match, does not contain, does not start with or does not end with. Moreover, they can be case-insensitive by adding 'i' at the beginning like 'iexact', etc. The full list of tag matching functions is obtained by invoking `sppasTagCompare().get_function_names()`.

The next examples illustrate how to work with such pattern matching filter. In this example, `f1` is a filter used to get all phonemes with the exact label 'a'. On the other side, `f2` is a filter that ignores all phonemes matching with 'a' (mentioned by the symbol '~') with a case insensitive comparison (iexact means insensitive-exact).

```
tier = trs.find("PhonAlign")
f = sppasFilter(tier)
ann_set_a = f.tag(exact='a')
ann_set_aA = f.tag(iexact='a')
```

The next example illustrates how to write a complex request. Notice that `r1` is equal to `r2`, but getting `r1` is faster:

```
tier = trs.find("TokensAlign")
f = sppasFilter(tier)
r1 = f.tag(startswith="pa", not_endswith='a', logic_bool="and")
r2 = f.tag(startswith="pa") & f.tag(not_endswith='a')
```

With this notation in hands, it is easy to formulate queries like for example: *Extract words starting by “ch” or “sh”*:

```
result = f.tag(startswith="ch") | f.tag(startswith="sh")
```

Practice:: Write a script to extract phonemes /a/ then phonemes /a/, /e/, /A/ and /E/. (solution: `ex15_annotation_label_filter.py`).

Filter on the duration

The following matching names are proposed to select annotations:

- ‘lt’ means that the duration of the annotation is lower than the given one;
- ‘le’ means that the duration of the annotation is lower or equal than the given one;
- ‘gt’ means that the duration of the annotation is greater than the given one;
- ‘ge’ means that the duration of the annotation is greater or equal than the given one;
- ‘eq’ means that the duration of the annotation is equal to the given one;
- ‘ne’ means that the duration of the annotation is not equal to the given one.

The full list of duration matching functions is obtained by invoking `sppasDurationCompare().get_function_names`

Next example shows how to get phonemes during between 30 ms and 70 ms. Notice that `r1` and `r2` are equals!

```
tier = trs.find("PhonAlign")
f = sppasFilter(tier)
r1 = f.dur(ge=0.03) & f.dur(le=0.07)
r2 = f.dur(ge=0.03, le=0.07, logic_bool="and")
```

Practice: Extract phonemes ‘a’ or ‘e’ during more than 100ms (solution: `ex16_annotation_dur_filter.py`).

Filter on position in time

The following matching names are proposed to select annotations:

- `rangefrom` allows to fix the begin time value,
- `rangeto` allows to fix the end time value.

Next example allows to extract phonemes ‘a’ of the 5 first seconds:

```
tier = trs.find("PhonAlign")
f = sppasFilter(tier)
result = f.tag(exact='a') & f.loc(rangefrom=0., rangeto=5., logic_bool="and")
```

Creating a relation function

Relations between annotations is crucial if we want to extract multimodal data. The aim here is to select intervals of a tier depending on what is represented in another tier.

James Allen, in 1983, proposed an algebraic framework named Interval Algebra (IA), for qualitative reasoning with time intervals where the binary relationship between a pair of intervals is represented by a subset of 13 atomic relation, that are:

- distinct because no pair of definite intervals can be related by more than one of the relationships;
- exhaustive because any pair of definite intervals are described by one of the relations;
- qualitative (rather than quantitative) because no numeric time spans are considered.

These relations and the operations on them form “Allen’s Interval Algebra”.

Pujari, Kumari and Sattar proposed INDU in 1999: an Interval & Duration network. They extended the IA to model qualitative information about intervals and durations in a single binary constraint network. Duration relations are: greater, lower and equal. INDU comprises of 25 basic relations between a pair of two intervals.

`anndata` implements the 13 Allen interval relations: before, after, meets, met by, overlaps, overlapped by, starts, started by, finishes, finished by, contains, during and equals; and it also contains the relations proposed in the INDU model. The full list of matching functions is obtained by invoking `sppasIntervalCompare().get_function_names()`.

Moreover, in the implementation of `anndata`, some functions accept options:

- before and after accept a `max_delay` value,
- overlaps and overlappedby accept an `overlap_min` value and a boolean `percent` which defines whether the value is absolute or is a percentage.

The next example returns monosyllabic tokens and tokens that are overlapping a syllable (only if the overlap is during more than 40 ms):

```
tier = trs.find("TokensAlign")
other_tier = trs.find("Syllables")
f = sppasFilter(tier)
f.rel(other_tier, "equals", "overlaps", "overlappedby", min_overlap=0.04)
```

Below is another example of implementing a request. *Which syllables stretch across 2 words?*

```
1 # Get tiers from a sppasTranscription object
2 tier_syll = trs.find("Syllables")
3 tier_toks = trs.find("TokensAlign")
4 f = sppasFilter(tier_syll)
5
6 # Apply the filter with the relation function
7 ann_set = f.rel(tier_toks, "overlaps", "overlappedby")
8
9 # To convert filtered data into a tier:
10 tier = ann_set.to_tier("SyllStretch")
```

Practice 1: Create a script to get tokens followed by a silence. (solution: `ex17_annotations_relation_filter1.py`).

Practice 2: Create a script to get tokens preceded by OR followed by a silence. (solution: `ex17_annotations_relation_filter2.py`).

Practice 3: Create a script to get tokens preceded by AND followed by a silence. (solution: `ex17_annotations_relation_filter3.py`).

6.6 More with SPPAS...

In addition to *anndata*, SPPAS contains several other API. They are all free and open source Python libraries, with a documentation and a set of tests.

Among others:

- *audiodata* to manage digital audio data: load, get information, extract channels, re-sample, search for silences, mix channels, etc.
- *calculus* to perform some math on data, including descriptive statistics.
- *resources* to access and manage linguistic resources like lexicons, dictionaries, etc.

References

7.1 References

7.1.1 How to cite SPPAS?

For a general citation of SPPAS, simply use the reference below. A PDF version of this publication is available in the folder `documentation` of the package.

Brigitte Bigi (2015). SPPAS - Multi-lingual Approaches to the Automatic Annotation of Speech. In “the Phonetician” - International Society of Phonetic Sciences, ISSN 0741-6164, Number 111-112 / 2015-I-II, pages 54-69.

For a specific purpose of SPPAS, like for automatic syllabification, forced-alignment, or for the request system, please refer to the related specific publication. All the papers are available on [the author website](http://www.lpl-aix.fr/~bigi/publications.html), at the following URL: <http://www.lpl-aix.fr/~bigi/publications.html>

7.1.2 SPPAS software description

Brigitte Bigi (2012). SPPAS: a tool for the phonetic segmentations of Speech, The eight international conference on Language Resources and Evaluation, Istanbul (Turkey), pages 1748-1755, ISBN 978-2-9517408-7-7.

Brigitte Bigi, Daniel Hirst (2012) SPeech Phonetization Alignment and Syllabification (SPPAS): a tool for the automatic analysis of speech prosody, Speech Prosody, Tongji University Press, ISBN 978-7-5608-4869-3, pages 19-22, Shanghai (China).

Brigitte Bigi, Daniel Hirst (2013). What's new in SPPAS 1.5?, Tools and Resources for the Analysis of Speech Prosody, Aix-en-Provence, France, pp. 62-65.

7.1.3 About linguistic resources

Brigitte Bigi, Bernard Caron, Abiola S. Oyelere (2017). **Developing Resources for Automated Speech Processing of the African Language Naija (Nigerian Pidgin)**. In 8th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 441-445, Poznań, Poland.

Mélanie Lancien, Marie-Hélène Côté, Brigitte Bigi (2020). **Developing Resources for Automated Speech Processing of Quebec French**. In Proceedings of The 12th Language Resources and Evaluation Conference, pp. 5323–5328, Marseille, France.

Brigitte Bigi, Abiola S. Oyelere, Bernard Caron (submitted). **Resources for Automated Speech Segmentation of the African Language Naija (Nigerian Pidgin)**. Human Language Technologies Challenges for Computer Science and Linguistics LNAI, Springer, Heidelberg.

7.1.4 About Search for IPUs

Brigitte Bigi, Béatrice Priego-Valverde (2019). **Search for Inter-Pausal Units: application to Cheese! corpus**. In 9th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 289-293, Poznań, Poland.

7.1.5 About Text Normalization

Brigitte Bigi (2011). **A Multilingual Text Normalization Approach**, 2nd Less-Resourced Languages workshop, 5th Language & Technology Conference, Poznan (Poland).

Brigitte Bigi (2014). **A Multilingual Text Normalization Approach**, Human Language Technologies Challenges for Computer Science and Linguistics LNAI 8387, Springer, Heidelberg. ISBN: 978-3-319-14120-6. Pages 515-526.

Roxana Fung, Brigitte Bigi* (2015). **Automatic Word Segmentation for Spoken Cantonese**, The Oriental Chapter of COCOSDA (International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques).

7.1.6 About Phonetization

Brigitte Bigi, Pauline Péri, Roxane Bertrand (2012). **Orthographic Transcription: Which Enrichment is required for Phonetization?**, Language Resources and Evaluation Conference, Istanbul (Turkey), pages 1756-1763, ISBN 978-2-9517408-7-7.

Brigitte Bigi (2013). **A phonetization approach for the forced-alignment task**, 3rd Less-Resourced Languages workshop, 6th Language & Technology Conference, Poznan (Poland).

Brigitte Bigi (2016). **A phonetization approach for the forced-alignment task in SPPAS**, Human Language Technologies Challenges for Computer Science and Linguistics, LNAI 9561, , pp. 515–526, Springer, Heidelberg. ISBN: 978-3-319-14120-6.

7.1.7 About Forced-Alignment

Brigitte Bigi (2012). **The SPPAS participation to Evalita 2011**, Working Notes of EVALITA 2011, Rome (Italy), ISSN: 2240-5186.

Brigitte Bigi (2014). **Automatic Speech Segmentation of French: Corpus Adaptation**. 2nd Asian Pacific Corpus Linguistics Conference, p. 32, Hong Kong.

Brigitte Bigi (2014). **The SPPAS participation to Evalita 2014**, Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 and the Fourth International Workshop EVALITA 2014, Pisa (Italy). Editors R. Basili, A. Lenci, B. Magnini. ISBN 978-886741-472-7. Volume 2. Pages 127-130.

Brigitte Bigi, Christine Meunier (2018). **euh, rire et bruits en parole spontanée : application à l'alignement forcé**, Actes des 32èmes Journées d'études sur la Parole, Aix-en-Provence, France.

Brigitte Bigi, Christine Meunier (2018). **Automatic speech segmentation of spontaneous speech**. Revista de Estudos da Linguagem. International Thematic Issue: Speech Segmentation. Editors: Tommaso Raso, Heliana Mello, Plinio Barbosa, Volume 26, number 4, pages 1489-1530, e-ISSN 2237-2083.

7.1.8 About Syllabification

Brigitte Bigi, Christine Meunier, Irina Nesterenko, Roxane Bertrand (2010). **Automatic detection of syllable boundaries in spontaneous speech**, Language Resource and Evaluation Conference, pages 3285-3292, La Valetta, Malte.

Brigitte Bigi, Caterina Petrone, Leonardo Lancia (2014). **Automatic Syllabification of Italian: adaptation from French**. Laboratory Approaches to Romance Phonology VII, Aix-en-Provence (France).

Brigitte Bigi, Caterina Petrone (2014). **A generic tool for the automatic syllabification of Italian**, Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 and the Fourth International Workshop EVALITA 2014, Pisa (Italy). Editors R. Basili, A. Lenci, B. Magnini. ISBN 978-886741-472-7. Volume 1. Pages 73-77.

Brigitte Bigi, Katarzyna Klessa (2015). **Automatic Syllabification of Polish**, 7th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 262–266, Poznan, Poland.

7.1.9 About Repetitions

Brigitte Bigi, Roxane Bertrand, Mathilde Guardiola (2014). **Automatic detection of other-repetition occurrences: application to French conversational speech**, 9th International conference on Language Resources and Evaluation (LREC), Reykjavik (Iceland), pp. 2648-2652. ISBN: 978-2-9517408-8-4.

7.1.10 About analyses tools

Brigitte Bigi (2019). **Filtering multi-levels annotated data**. In 9th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 13-14, Poznań, Poland.

Brigitte Bigi, Jorane Saubesty (2015). **Searching and retrieving multi-levels annotated data**, Proceedings of Gesture and Speech in Interaction, Nantes (France).

7.1.11 About the API

Brigitte Bigi, Tatsuya Watanabe, Laurent Prévot (2014). **Representing Multimodal Linguistics Annotated Data**, 9th International conference on Language Resources and Evaluation (LREC), Reykjavik (Iceland), pages 3386-3392. ISBN: 978-2-9517408-8-4.

Brigitte Bigi (2018). **Annotation representation and file conversion tool**. Contributi del Centro Linceo Interdisciplinare ‘Beniamino Segre’ ISSN 0394-0705), 137, pp. 99-116.

7.1.12 Related references

Some results of analyses

Marion Tellier, Gale Stam, Brigitte Bigi (2012). **Same speech, different gestures?**, 5th International Society for Gesture Studies (ISGS), Lund, Sweden.

Marion Tellier, Gale Stam, Brigitte Bigi (2013). **Gesturing While Pausing In Conversation: Self-oriented Or Partner-oriented?**, TIGER, Tillburg.

Laurent Prévot, Brigitte Bigi, Roxane Bertrand (2013). **A quantitative view of feedback lexical markers in conversational French**, 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Metz, France.

Brigitte Bigi, Katarzyna Klessa, Laurianne Georgeton, Christine Meunier (2015). **A syllable-based analysis of speech temporal organization: a comparison between speaking styles in dysarthric and healthy populations**. Interspeech2015, Dresden (Germany), pages 2977-2981.

Laurent Prevot, Jan Gorisch, Roxane Bertrand, Emilien Gorene and Brigitte Bigi (2015). **A SIP of CoFee: A Sample of Interesting Productions of Conversational Feedback**, 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Prague (Czech Republic), pp. 149–153.

Brigitte Bigi, Roxane Bertrand (2016). **Laughter in French Spontaneous Conversational Dialogs**, Tenth International Conference on Language Resources and Evaluation (LREC 2016), pp. 2168-2174, Portoroz, Slovenia.

Béatrice Priego-Valverde, Brigitte Bigi, Salvatore Attardo, Lucy Pickering, Elisa Gironzetti (2018). **Is smiling during humor so obvious? A cross-cultural comparison of smiling behavior in humorous sequences in American English and French interactions**. Intercultural Pragmatics. 10/2018; 15(4):563-591.

Some corpora

Sophie Herment, Anastasia Loukina, Anne Tortel, Daniel Hirst, Brigitte Bigi (2012). **AixOx, a multi-layered learners corpus: automatic annotation**, Proceedings of international conference on corpus linguistics, Jaën (Spain).

Ellen Gurman Bard, Corine Astésano, Alice Turk, Mariapaola D'imperio, Noel Nguyen, Laurent Prévot, Brigitte Bigi (2013). **Aix MapTask: A (rather) new French resource for prosodic and discourse studies**, Proceedings of Tools and Resources for the Analysis of Speech Prosody, Aix-en-Provence, France, Eds B. Bigi and D. Hirst, ISBN: 978-2-7466-6443-2, pp. 15-19.

Daniel Hirst, Brigitte Bigi, Hyongsil Cho, Hongwei Ding, Sophie Herment, Ting Wang (2013). **Building OMProDat: an open multilingual prosodic database**, Proceedings of Tools and Resources for the Analysis of Speech Prosody, Aix-en-Provence, France, Eds B. Bigi and D. Hirst, ISBN: 978-2-7466-6443-2, pp. 11-14.

Jan Gorish, Corine Astésano, Ellen Gurman Bard, Brigitte Bigi, Laurent Prévot (2014). **Aix Map Task corpus: The French multimodal corpus of task-oriented dialogue**, 9th International conference on Language Resources and Evaluation (LREC), Reykjavik (Iceland), pages 2648-2652. ISBN: 978-2-9517408-8-4.

Christine Meunier, Cécile Fougeron, Corinne Fredouille, Brigitte Bigi, Lise Crevier-Buchman, Elisabeth Delais-Roussarie, Laurianne Georgeton, Alain Ghio, Imed Laaridh, Thierry Legou, Claire Pillot-Loiseau, Gilles Pouchoulin (2016). **The TYPALOC Corpus: A Collection of Various Dysarthric Speech Recordings in Read and Spontaneous Styles**, Tenth International Conference on Language Resources and Evaluation (LREC 2016), pp. 4658-4665, Portoroz, Slovenia.

Béatrice Priego-Valverde, Brigitte Bigi, Mary Amoyal (2020). **“Cheese!”: a Corpus of Face-to-face French Interactions. A Case Study for Analyzing Smiling and Conversational Humor**. In Proceedings of The 12th Language Resources and Evaluation Conference, pp. 467–475, Marseille, France.

Birgit Rauchbauer, Youssef Hmamouche, Brigitte Bigi, Laurent Prévot, Magalie Ochs, Thierry Chaminade (2020). **Multimodal Corpus of Bidirectional Conversation of Human-human and Human-robot Interaction during fMRI Scanning**. In Proceedings of The 12th Language Resources and Evaluation Conference, pp. 668–675, Marseille, France.

Tools re-implemented into SPPAS

Daniel Hirst and Robert Espesser (1993). *Automatic modelling of fundamental frequency using a quadratic spline function*. Travaux de l'Institut de Phonétique d'Aix. vol. 15, pages 71-85.

D.-J. Hirst (2011). *The analysis by synthesis of speech melody: from data to models*, Journal of Speech Sciences, vol. 1(1), pages 55-83.

Dafydd Gibbon (2013). **TGA: a web tool for Time Group Analysis**, Tools and Resources for the Analysis of Speech Prosody, Aix-en-Provence, France, pp. 66-69.

7.2 SPPAS in research projects

You used SPPAS into your research project? Send an e-mail to the author to append it into this section!

7.2.1 MULTIPHONIA

SPPAS was used to annotate the MULTIPHONIA corpus (MULTImodal database of PHONetics teaching methods in classroom InterActions) created by Charlotte ALAZARD, Corine ASTESANO, Michel BILLIÈRES.

This database consists of audio-video classroom recording comparing two methods of phonetic correction (the “traditional” articulatory method, and the Verbo-Tonal Method). This database is composed of 96 hours of pronunciation classes with beginners and advanced students of French as a Foreign Language. Every class lasted approximatively 90 minutes. This multimodal database constitutes an important resource for Second Language Acquisition’s researchers. Hence, MULTIPHONIA will be enriched at many different levels, to allow for segmental, prosodic, morphosyntactic, syntactic, lexical and gestural analyses of L2 speech.

Charlotte Alazard, Corine Astésano, Michel Billières **MULTIPHONIA: a MULTImodal database of PHONetics teaching methods in classroom InterActions**, Language Resources and Evaluation Conference, Istanbul (Turkey), May 2012.

MULTIPHONIA: <http://www.sldr.org/sldr000780/en>

7.2.2 Amennpro

SPPAS was used for the annotation of the French part of the AixOx corpus.

- four way recordings of French and English texts;
- read by English and French speakers;
- non-native speakers were divided into advanced and beginners.

Download the AixOx corpus: <http://www.sldr.fr/sldr000784/>

Remark:

Some examples are located in the samples-fra directory (files F_F_*.*) and in the samples-eng (files E_E_*.*) .

7.2.3 Evalita 2011: Italian phonetization and alignment

Evalita 2011 was the third evaluation campaign of Natural Language Processing and Speech tools for Italian, supported by the NLP working group of AI*IA (Associazione Italiana per l’Intelligenza Artificiale/Italian Association for Artificial Intelligence) and AISV (Associazione Italiana di Scienze della Voce/Italian Association of Speech Science).

SPPAS participated to the Forced Alignment on Spontaneous Speech for both tasks:

- Phone segmentation

- Word segmentation

The corpus was a set of Dialogues, map-tasks:

- 3h30 speech;
- 15% phones are: “sil”, filled-pauses, garbage.

7.2.4 Cofee: Conversational Feedback

In a conversation, feedback is mostly performed through short utterances produced by another participant than the main current speaker. These utterances are among the most frequent in conversational data. They are also considered as crucial communicative tools for achieving coordination in dialogue. They have been the topic of various descriptive studies and often given a central role in applications such as dialogue systems. Cofee project addresses this issue from a linguistic viewpoint and combines fine-grained corpus analyses of semi- controlled data with formal and statistical modelling.

Cofee is managed by Laurent Prévot: <http://cofee.hypotheses.org/>

Cofee corpora and the use of SPPAS on such corpora is presented in:

Jan Gorish, Corine Astésano, Ellen Gurman Bard, Brigitte Bigi, Laurent Prévot (2014). *Aix Map Task corpus: The French multimodal corpus of task-oriented dialogue*, 9th International conference on Language Resources and Evaluation (LREC), Reykjavik (Iceland).

7.2.5 Variamu: Variations in Action: a MULTilingual approach

Variamu is an international collaborative joint project co-funded by Amidex. The scientific object of the collaboration is the issue of language variation addressed from a comparative perspective. Speech and language knowledge supports a growing number of strategic domains such as Human Language Technologies (HLT), Language Learning (LL), and Clinical Linguistics (CL). A crucial issue to these domains is language variation, which can result from dysfunction, proficiency, dialectal specificities, communicative contexts or even inter-individual differences. Variation is often an important part of the question itself.

This network will be structured around 3 research axes, all centered on the variation concept:

1. Language technologies,
2. Linguistics and Phonetics,
3. Speech and Language Pathologies.

SPPAS is mainly concerned by the first axe.

The first result of this project is the participation of SPPAS at the Evalita 2014 campaign, for the FACS task: Forced-Alignment on Children Speech.

The second result of this project is the support of Cantonese into SPPAS, thanks to a collaboration with Prof. Tan Lee of the University of Hong Kong.

SPPAS Release notes

8.1 The early versions

Versions 1.0 to 1.3 was made only of `tcsh` and `gawk` scripts. It was developed under Linux system and was efficiently tested under Windows with Cygwin.

8.1.1 Version 1.0

(2011, 9th March)

The only feature was that it was able to perform speech segmentation of English read speech.

8.1.2 Version 1.1

(2011, 7th June)

This was mainly the debug of the previous version and some code re-organization and cleaning.

8.1.3 Version 1.2

(2011, 23th July)

The support of English, French and Italian was added: a lexicon, a pronunciation dictionary and an acoustic model of each language was created. Three annotations were implemented: Tokenization, Phonetization, Alignment.

The basis for a *multi-lingual methodology* was already there.

8.1.4 Version 1.3

(2011, 12th December)

This is a transitional version, from the scripts language to Python programming language. The main fixes and improvements were:

Development:

- MacOS support
- bugs corrected in many scripts
- `check.csh` -> `check.py`
- `sppas.csh` -> `sppas.py`
- GUI: `zenity` -> `pyGtk`
- `sox` limited to the use of re-sampling audio signal
- a python library to manage wav files is added.
- IPU's segmentation automatic annotation

Resources:

- Italian dictionary improved
- Italian acoustic model changed (triphones trained from map-task dialogues)
- French acoustic model changed

But the program still haven't a name and isn't distributed.

8.2 The birth of SPPAS

8.2.1 SPPAS 1.4.0

(2012, 14th June)

It's the official birth of SPPAS: the software has a name, a license, a GUI, a web-page and is freely distributed to the community.

The source code is only based on Python 2.7 language and the GUI is based on WxPython 2.8.x or 2.9.x (on MacOS, wxpython must be 32 bits). SPPAS requires also `sox` and `julius` software to be installed.

Automatic annotations:

- add Momel: Momel (MOdelling MELody) is an algorithm developed by Daniel Hirst and Robert Espesser for the analysis and synthesis of intonation patterns. Momel needs a .hz files with pitch values: ASCII file with one value each 10ms. This Momel implementation can be used directly, with a large set of options: `> python $SPPAS/scripts/lib/momel.py -h`
or in SPPAS, using the GUI, or in text-mode (only with default options): `> $SPPAS/sppas.py -i INPUT -momel`
- add the first version of INTSINT, proposed by Daniel Hirst. INTSINT is an acronym for International Transcription System for INTonation. INTSINT codes the intonation of an utterance by means of an alphabet of 8 discrete symbols constituting a surface phonological representation of the intonation.

Packaging:

- merge AUTHORS, VERSION and README files in the README file

- create packages in the lib directory (2012-03-20)
- ipu-segmentation in python (2012-02-16)
- improve ipu segmentation algorithm (2012-05-24)
- phonetization in python (2012-02-27), with an unknown word phonetization.
- alignment entirely in python (2012-03-29)
- syllabification implemented with python, and tool/syllabify.py
- add a simple terminal controller (not needed for Unix, just for “DOS”)
- Momel can read PitchTier (from Praat) files
- Momel can deal with long sounds (not only one IPU)
- manage source files, comments, exceptions...
- SPPAS works as a single instance by mean of a lock file.
- SPPAS can be installed in a directory containing spaces and can deal with file names with spaces (bug fixed: 2012-06-11)

GUI:

- GUI with the wxpython library (2012-03-27)
- Manage a list of selected files
- Add a “File information”
- Add a “Wav player”
- Fix options to each annotation step with the menu
- Open the log file after each process

Resources:

- French dictionary is using UTF-8 (instead of iso8859-1 in previous versions)
- French dictionary is based on X-SAMPA phone set
- Chinese: work with chinese characters instead of pinyin.

Known Bugs:

- The wav player can not play wav files if the filename contains ‘#’.
- The first line of the Italian dictionary must be changed to “# [#] #”.

8.2.2 SPPAS 1.4.1

(2012, 13th July)

Resources:

- Updated English acoustic model (from voxforge, 2012-06-25)
- English acoustic models converted to X-SAMPA

Automatic annotations:

- IPU's Segmentation annotation performs a simple silence detection if no transcription is available (the volume is automatically adjusted)

- A specific language can be selected for each annotation depending on available resources
- Updated transcription conventions:
 - truncated words: a ‘-’ at the end of the token string (an ex- example)
 - liaisons: the ‘letter’ between ‘=’ (an =n= example)
 - noises: * (only for FR and IT)
 - short pauses: + (a + example)
 - silences: # (a # example)

GUI:

- Create (systematically) a merged annotations TextGrid file.

Development:

- Package’s management

8.2.3 SPPAS 1.4.2

(2012, 2nd August)

GUI:

- add a panel with a Transcription editor
- IPU segmentation can split a wav into tracks
- IPU segmentation can fix a shift value to boundaries
- IPU segmentation: min-volume option is removed (because the min-volume value is automatically adjusted)
- “File Information” button adds some tier manipulation tools: cut/copy/paste/rename/duplicate/move/preview/filter

Known bug:

- The filter frame is not working under Windows XP.

8.2.4 SPPAS 1.4.3

(2012, 10th October)

This is primarily a bug-fix release. The author is addressing many thanks to all users who send their comments!

GUI:

- Frames and Dialog design is more uniform.
- Users preferences changed. Themes and colors introduced.
- Help is available.

Automatic annotations:

- Bug fixed for phonetization/alignement if the input transcription contains series of silence intervals or series of speech intervals. Previous versions was supposing a **strict IPU**s input transcription.
- Tokenization is done.

Development:

- Code cleaning in the package wxGUI.
- Debug...

8.2.5 SPPAS 1.4.4

(2012, 6th December)

GUI:

- add information/options when “Request” a file
- debug Request/Filter (for Windows systems)

Automatic annotations:

- New Italian acoustic model
- New Chinese acoustic model, and some minor changes in the dictionary

Development:

- “.trs” files support (transcriptions from Transcriber)
- debug (alignment, tokenization)
- add “.lab” files export (HTK format)

Known bugs:

- Alignment: it fails under Windows, if `julius` is not installed properly.
- Syllabification: the last syllable of each file is “broken”.
- Alignment error if unknown words during phonetization.

8.2.6 SPPAS 1.4.5

(2013, 15th January)

Development:

- Correct a few bugs of the previous version (phonetization, alignment, syllabification)
- “.eaf” files support (transcriptions from Elan software)
- add script `tierfilter.py`
- add script `tiercombine.py`

Automatic annotations:

- Experimental version of Vietnamese
- add Tokenization as a specific annotation level
- add Phonetization of PinYin

GUI:

- Request/Filter a tier: multiple patterns filtering
- Request: add a “New File” button

8.2.7 SPPAS 1.4.6

(2013, 12th February)

GUI:

- Improved Request/Filter a tier:
 - add new modes: not contains, not starts with, not ends with
 - add time constraints: minimum duration, maximum duration, meets
 - multiple modes selection (replace radio buttons by check buttons)
- Add Requests/Stats to obtain basic statistics
- Requests Copy/Cut/Paste/Duplicate/Save debugged

Automatic annotations:

- IPU segmentation: can take a “Name” tier to fix names of tracks (if the “Split into tracks” option is checked)

8.2.8 SPPAS 1.4.7

(2013, 25th March)

Development:

- re-organization of lib, except for the wxGUI library.
- Cancel the sppas.lock file creation when SPPAS is running.
- Requests/Filter: “Starts search at...” and “Ends search at...”
- Requests/Filter: remove negative search (because of a bug...). Replaced by a “reverse” option.

Resources:

- add experimental version of Taiwanese automatic segmentation (from romanized transcriptions)

Annotations:

- New version of INTSINT, based on the algorithm proposed in (Hirst 2011) and implemented in the last version of Momel/INTSINT Praat plugin!

8.2.9 SPPAS 1.4.8

(2013, 30th May)

Development:

- reorganization of the GUI code. SPPAS is made of:
 - automatic annotations;
 - 3 components (wavplayer, transcriber, requests).
- sppas.py removed. sppas.bat created (for Windows).
- Input/Output library entirely changed

GUI:

- components (wavplayer, transcriber, requests) in separate frames
- new design
- Help and documentation changed, expanded and improved

8.2.10 SPPAS 1.4.9

(2013, 3rd July)

SPPAS has a new and more colored logo!

Development:

- bug correction:
 - Bug fixed in IPU segmentation with option “Split into tracks”
 - Bug fixed in Momel with some rare files
 - Bug fixed to create a merged file
 - Bug fixed in align and IPU seg.
 - Bug fixed in Transcriber
- library organization revised

GUI:

- Add an export button to the File List Panel
- Migrate wav infos from the Requests component to the Wav player component
- Volume control removed in the Wav Player component
- Save As improved in the Requests component

8.2.11 SPPAS 1.5.0

(2013, 2nd August)

Development:

- bug correction

Annotation:

- Tokenization: TOE support finalized

GUI:

- Transcribe: debug of IPU player, for MacOS

Resources:

- New French acoustic model. Attention: phoneset changed.
- New French dictionary: use the new phoneset!
- New Taiwanese acoustic model. Attention: phoneset changed: accept some chinese...
- New Taiwanese dictionary: use the new phoneset + some chinese syllables added.
- Vietnamese removed: due to the lack of data, the model can't be improved.

8.2.12 SPPAS 1.5.1

(2013, 29th August)

Development:

- bug correction in annotations
- help system debugged

Annotation:

- Phonetization of unknown words improved

Resources:

- French dict modified

8.2.13 SPPAS 1.5.2

(2013, 27th September)

All resources are moved into the “resources” directory.

Development:

- bug correction in annotations and GUI

Resources:

- French dict modified
- New resources for the tokenization

Components:

- “Statistics”: an avanced component to estimate (and save!) descriptive statistics on multiple annotated files

8.2.14 SPPAS 1.5.3

(2013, 25th October)

Resources:

- Add Spanish support

Components:

- improved Statistics component
- Add a “Filter” component, first version with a basic GUI

GUI:

- Help updated.

8.2.15 SPPAS 1.5.4

(2013, 3rd December)

Components:

- Wav Player changed.
- Information and Requests removed. Replaced by Data Roamer.
- Transcriber removed. Replaced by IPU Transcriber.
- Statistics updated.
- Filter updated.

Annotations:

- Add Repetitions (detection of sources only)

Notice that this is the first stable release.

8.2.16 SPPAS 1.5.5

(2013, 23th December)

Development:

- improved annotationdata (add methods: Find, Index; add uncertain label; debug radius).

Components:

- debug

GUI:

- Tips at start-up
- New theme: Christmast

8.2.17 SPPAS 1.5.6

(2014, 28th January)

Development:

- improved annotationdata (add methods: Near, Search).
- Package cleaning!

Components:

- debug

Resources:

- New Italian dictionary (including gemination)

8.2.18 SPPAS 1.5.7

(2014, 18th February)

Development:

- Plugins manager added.

Resources:

- Japanese support, thanks to the resources available on the Julius website.

8.2.19 SPPAS 1.5.8

(2014, 18th March)

Development:

- annotationdata: more flexibility while adding annotations, add sub-divisions, export csv modified, docstrings, import/export in a native format (xra, version 1.0).

Documentation:

- add a PDF file of slides: “SPPAS for dummies”

8.2.20 SPPAS 1.5.9

(2014, 15th April)

Components:

- new: DataViewer, experimental version

Annotations:

- syllabification rules: accept 2 types of vowels (V and W)
- syllabification: faster!

Development:

- Export Elan
- Import/Export xra. XRA is the native format of SPPAS annotation files. See etc/xra for details.

Resources:

- New French acoustic model

8.2.21 SPPAS 1.6.0

(2014, 22th May)

Package:

- Rename folder “Doc” to “documentation”
- A documentation is created
- SPPAS-for-dummies updated

Development:

- helpsystem removed
- GUI: package re-organization, re-implementation.
- Alignment: choice of the aligner (julius, hvite or basic)

Resources:

- new acoustic model: FR-Read and FR
- new acoustic model: ZH
- new acoustic model: TW
- new acoustic model: SP

8.2.22 SPPAS 1.6.1

(2014, 26th September)

Development:

- bug correction (export, syllabification, alignment)
- DataViewer, major bugs corrected.

Resources:

- Italian: new pronunciation dictionary and new acoustic model
- add resources for Catalan

GUI:

- add a new Export button

8.2.23 SPPAS 1.6.2

(2014, 21th October)

Resources:

- language names changed. They are now corresponding to the international standard ISO639-3. See <http://www-01.sil.org/iso639-3/>
- Mandarin Chinese dictionary changed.
- Catalan dictionary changed.

8.2.24 SPPAS 1.6.3

(2014, 2nd November)

Resources:

- Add resources for Cantonese

Documentation:

- It's now (more or less) finished!

Development:

- Support of Elan improved (add Controlled vocabulary)

GUI:

- Change the organization of the main frame: annotations above components

This version is known to be a stable release.

8.3 The beginnings of SPPAS

8.3.1 SPPAS 1.6.4

(2014, 5th December)

From this version, SPPAS requires wxpython to be updated to version 3.0, particularly for MacOS users, and they need to install the 64 bits version. It is recommended to Windows users to install Python 2.7 and wxpython in 32bits.

Development:

- Package re-organized!
- Phonetization of unknown words improved
- Support of upper/lower of the extension of speech files (wav, WAV)
- Tokenization of languages with dictionaries in upper case (eng, ita): bug fixed.

- Creates systematically a dump file of resources for a faster load at the next use
- Read TextGrid files exported by Elan: bug fixed.
- `sppas.command` checks the system and run either in 32 or 64 bits (MacOS)

Components:

- IPUScribe replaces IPUTranscriber mainly for the support of large files: tested with a file of 1 hour speech (143 Go) and 800 IPUs.
- SndRoamer replaces WavPlayer
- Dataroamer has also a new version

8.3.2 SPPAS 1.6.5

(2014, 17th December)

This is primarily a bug-fix release.

Development:

- all programs in “bin” and “scripts” sub-directories were revised and tested, or removed.

Annotation:

- Tokenization: code cleaning and re-organisation.

GUI:

- Procedure outcome report: print a warning message in the log file if no file is matching the expected extension

8.3.3 SPPAS 1.6.6

(2015, 19th January)

Web site host has changed: <http://sldr.org/sldr00800/preview/>

Documentation completed and updated. Now, only the documentation of all the components is missing.

Annotations:

- log messages more explicit and status printed with intuitive colors.
- management of input/output file format re-done: now easier for the user.

Development:

- package architecture revised: mainly “sppasgui” and “components” merged in “wxgui”, and many other changes.
- thread removed in automatic annotation process.
- debug of alignment: if too short units.
- radius value properly fixed in most of the automatic annotations.

GUI:

- GUI is more homogeneous and pretty (hope!)
- Show the date in the status bar
- New Settings frame:
 - 4 icon themes available
 - Choice of foreground and background colours
 - Choice of the font
 - Choice of the input/output file format of annotations
- New in Help menu:
 - access to the project homepage
 - access to the online documentation
- New Feedback window
- New Help browser
- Add Keyboard shortcuts:
 - ALT+F4 to exit,
 - CTRL+A to add files in FLP
 - SHIFT+CTRL+A to add a directory in FLP
 - Del to remove selected files of the FLP
 - SHIFT+Del to erase selected files of the FLP
 - CTRL+C to copy files
 - CTRL+E to export files
 - F5 to refresh the FLP
 - F1 to open the help browser
 - F2 to open the “About” frame

Components:

- GUI design unified for DataRoamer, SndPlayer, IPUscribe and SppasEdit
- New Tier Preview frame (still under development)
- SndPlayer print information about a sound file with colors:
 - Green: the information corresponds to the SPPAS requirements

- Orange: the information does not exactly corresponds to the requirements however SPPAS is able to deal with (after conversion)
- Red: SPPAS does not support. It must be converted before using it!

8.3.4 SPPAS-1.6.7

(2015, 16th February)

Automatic Annotations:

- By default, tokenization produces only one tier. Check an option to get TokensStd and TokensFaked, in case of EOT.
- radius value properly fixed in most of the automatic annotations.

GUI:

- Tested and debugged on MacOS (version 10.9.5, with wxpython 3.0.2)

Development:

- Tier hierarchy partly implemented: TimeAlignement and TimeAssociation are two “links” that can be fixed between tiers.

Annotations:

- Add Polish support

8.3.5 SPPAS-1.6.8

(2015, 9th April)

Resources:

- new French acoustic model
- new English acoustic model (VoxForge nightly build of March, 15th, 2015)
- add phoneset mapping tables

Development:

- Add a phoneme mapping in models, to allow both the dictionary to include real X-SAMPA symbols and the acoustic model to be compatible with Hvite requirements (only ASCII).
- annotationdata bug correction with min and max values
- IPU's Segmentation:

- bug correction when split into tracks with a tier “Name”
 - add the ipu number in speech segments if silence/speech segmentation
- Self-repetitions debug in finding the repetition interval

GUI:

- DataRoamer: “New” button debugged
- DataRoamer: Add a button “Radius” to adjust manually the vagueness of each bounday of a tier

8.4 The development phase

8.4.1 SPPAS-1.6.9

(2015, 14th May)

The installation of dependencies is simplified: `sox` is unnecessary. Python 2.7.x, WxPython and Julius are the only remaining dependencies.

Development:

- package `annotationdata.filter` updated to support last changes in `annotationdata`: multiple labels and numerical labels.
- package `annotationdata.io`:
 - `praat.py` newly created. Support of `TextGrid`, `PitchTier` and `IntensityTier` files completely re-written
 - `htk.py` newly created to support `.lab` and `.mlf` files
 - `sc lite.py` newly created to support `.stm` and `.ctm` files
 - `signaix.py` newly created to support `.hz` files
 - SPPAS native format XRA upgraded to version 1.1 to support improvements of the library.
- package `annotationdata`:
 - updated object `Transcription` to support more/different input data
 - updated hierarchy
 - updated meta-data
- package `signal`: partially re-written. As a consequence, `sox` won’t be neither used, but the file conversion (if required) is slower.

GUI:

- `DataFilter` component partially re-written to facilitate its use
- Preview frame modified

8.4.2 SPPAS-1.7.0

(2015, 3th July)

Development:

- package annotationdata.io:
 - add support of subtitles: sub, srt
 - elan.py created to replace eaf.py: allows a full support of Elan annotations
 - transcriber.py created to replace trs.py for a better support of Transcriber files
 - anvil.py allows to import anvil files
- package annotationdata:
 - updated hierarchy (simplified)
 - updated meta-data
- package signal:
 - support of audio files re-written: can open/save wav, aiff and au files
 - add a lot of possibilities to manage one channel of an audio file

GUI:

- DataFilter component finalized: can deal with alternative labels, and typed labels (string, number, boolean)
- Statistics component fully re-written to facilitate its use
- SndRoamer displays more properties of audio files (min, mean and max added)

Annotations:

- IPUs Segmentation produces 2 tiers if a transcription is given: the IPUs segmentation itself, and the Transcription time-aligned at the IPUs level.

8.4.3 SPPAS-1.7.1

(2015, 5th August)

Development:

- package re-organization:
 - package signal is now standalone
 - package resources is now standalone
 - package presenters created
- updated statistics estimations
- package annotationdata:

- add Media object
 - add CtrlVocab object
 - add inheritance of MetaObject for Annotation
- package annotationdata.io:
 - full debug of all file formats
 - add comments and documentation
 - add also some tests
 - add Media/CtrlVocab in some file formats
 - add Annotation Pro support of antx files (in API)

Components:

- add Time Group Analyser (TGA) in Statistics
- add Kappa estimation on labels of 2 tiers with the same number of intervals

Annotations:

- New version of XRA: 1.2
- Make XRA the default input/output file format

8.4.4 SPPAS-1.7.2

(2015, 3th September)

Development:

- updated XRA reader/writer to solve problems with upper/lower cases of elements
- updated Elan reader to be compatible with format 2.8
- updated Praat reader/writer for single/double quotes
- support of AnnotationPro antx files in the GUI
- add the julius score in PhonAlign annotation (can be seen only if XRA)
- remove tk dependency

8.4.5 SPPAS-1.7.3

(2015, 9th October)

Resources:

- New word-based vocab for Cantonese
- New word-based dict for Cantonese
- New phoneme-based acoustic model for Cantonese

Development:

- Descriptive statistics debugged for detailed panels.

8.4.6 SPPAS-1.7.4

(2015, 6th November)

Resources:

- Add a vocab for Portuguese
- Add a dict for Portuguese
- Add an acoustic model for Portuguese. It has to be noticed that it was constructed from French/Spanish/Italian models, not trained from data.

Samples:

- Add Portuguese
- Change some samples in various languages

Development:

- Debug of the Tokenizer.

8.4.7 SPPAS-1.7.5

(2015, 11th December)

Development:

- Add vagueness to the annotation duration estimation
- Bug correction while saving the procedure outcome report in the GUI

GUI:

- Changed all pictures to remove the problem with colorset of some of them
- Add a christmas theme for the pictures of tips
- IPUScribe improvements:
 - add keyboard shortcuts to play/pause/stop the sound
 - add a new button to auto-play the sound

8.4.8 SPPAS-1.7.6

(2016, 28th January)

Web site host has changed: <http://www.sppas.org/>

Development:

- IPU segmentation:
 - bug correction with option “split into tracks”
 - new option to add or not add the IPU index of each IPU into the transcription tier
- Tokenization:
 - bug correction in input tier selection
 - bug correction for replacements

Resources:

- add a vocabulary of Korean
- add an acoustic model for Korean (made from the English and the Taiwanese ones).

GUI:

- DataRoamer: bug correction of “Duplicate”

Others:

- Add a sample in Korean
- Updated references to cite SPPAS in publications

8.4.9 SPPAS-1.7.7

(2016, 30th March)

Resources:

- Correction of errors in most of the acoustics models, for /@@/ and /dummy/ models
- New vocabulary and pronunciation dictionary for Mandarin Chinese.

GUI:

- Dialogs are customized
- DataRoamer debug and changes: save buttons moved in the main toolbar.
- HelpSystem is (welcome) back!

Development:

- Add the API and a script to train acoustic models with HTK.
- Add Kullback-Leibler distance estimator.
- Add a script to compare segmentation of 2 tiers.
- Classes of the package resources are debugged, improved and extended.
- Alignment: bug correction if input starts by an empty interval.
- `sppas.command` modified for MacOS-X (use python if python2 not available)
- GUIs in `bin` directory are updated (test of python, wxpython, and so on).

8.4.10 SPPAS-1.7.8

(2016, 5th May)

Resources:

- Add a pronunciation mapping table `eng-fra.map`: to be used for the speech segmentation for French speakers reading an English text.

Development:

- Phonetization is extended: it can take into account a mapping table of phones to generate new pronunciation variants.
- Phonetization: use of minus instead of dots to separate phones as recommended in X-SAMPA standard.
- Alignment is restructured and extended: it can take into account two acoustics models and mix them.
- Filter is extended: Relations can be based on a delay between the intervals
- `annotationdata`: add Xtrans reader (extension: `.tdf`)
- Code cleaning in several packages.

GUI:

- Automatic annotations: more explicit log messages.

8.4.11 SPPAS-1.7.9

(2016, 3th June)

GUI:

- Some debug in `SppasEdit`
- `SndPlayer` renamed `AudioRoamer` because new functionalities were added:
 - See detailed information about each channel of an audio file
 - Can extract/save a channel or a fragment of channel
 - Can modify the frame rate and the sample width of a channel

- Can add a bias on amplitude values of a channel
- Can multiply amplitude values of a channel
- Can remove the offset of amplitude values of a channel

Automatic annotations:

- IPUs Segmentation fully re-implemented.
 - Silence/speech segmentation improved for both quality and fastness
 - Package code cleaning and re-organization.

8.5 The stabilization phase

8.5.1 SPPAS-1.8.0

(2016, 30th August)

GUI:

- Design fully revisited and tested under Linux Mint, Windows 10 and MacOS 10.9

Development:

- SLM package created: can estimate a statistical language model (without smooth method) on a small corpus

Automatic annotations:

- Add a diagnosis of files
- Tokenize extended: applied also on alternative labels
- Phonetize extended: applied also on alternative labels
- Alignment code cleaning and partly re-implemented
- Add “Chunk alignment”
- Use of a .ini file to configure each annotation instead of a sppas.conf file

8.5.2 SPPAS-1.8.1

(2016, 28th November)

A few tutorials are available on the web site.

Automatic annotations:

- Align: an ActivityDuration tier can be optionally added.
- Support of 3-columns tab-delimited files with .txt extension. It allows the compatibility with Audacity Label track files.
- Acoustic models training validated.

Resources:

- Catalan: new pronunciation dictionary and new acoustic model.

8.5.3 SPPAS-1.8.2

(2017, 18th January)

Analysis:

- debug of DataFilter

Resources:

- French vocabulary and dictionary updated

Development:

- new plugins package with a new plugin manager
- GUI integration of this new plugins system
- some unittest appended and all existing ones updated
- annotationdata.io renamed annotationdata.aio
- docstrings of some packages converted from epytext to reST syntax

GUI:

- DataStats, DataFilter and DataRoamer toolbars don't scroll anymore
- Themes management changed.
- Main font is managed by the Themes.

8.5.4 SPPAS-1.8.3

(2017, 10th March)

Development:

- Elan reader highly improved (faster reader).
- updated plugins

8.5.5 SPPAS 1.8.4

(2017, 10th April)

Development:

- Elan writer modified: create a time slot for each localization.

8.5.6 SPPAS 1.8.5

(2017, 20th April)

Development:

- Vizualizer renamed Visualizer
- AudioRoamer: bug with an icon corrected
- New Phonedit mrk format support.
- Updated AnnotationPro Antx reader/writer

8.5.7 SPPAS 1.8.6

(2017, 19th June)

Resources:

- Polish dictionary and acoustic model updated.

8.5.8 SPPAS 1.9.0

(2017, 28th July)

Programming:

- Relative imports used in the standard way for Python
- PEP 8 code style (except for wxgui/annotationdata)
- PEP 257 reST code documentation style (except for wxgui/annotationdata)
- Unittests:
 - existing tests verified, improved, extended
 - new tests added
 - tests migrated into the packages
- Compatibility for both Python 2.7 and Python > 3.2:
 - makeunicode.py contains functions and classes to deal with strings
 - utils, term, structs, resources, presenters, plugins, calculus packages: migration is done
- Exceptions are separately managed
- Introduction of a system for the internationalization of the messages. Done in English and French for the packages: audiodata, calculus, plugins, resources, structs, term, utils
- Package re-organization:

- new package “models”, with acm and slm
 - utils
 - resources
 - ...
- meta.py is replacing sp_glob.py
- new scripts: tieraligntophon.py, dictmerge.py
- new bin: pluginbuild.py
- Robustness to read malformed HTK-ASCII pronunciation dictionaries.
- Bug corrected in the management of pronunciation variants
- re-structured package TextNormalization
- re-structured package Repetitions
- new version of the plugins: updated and debugged.

Resources:

- Add support of Naija language (pcm)
- English-French mapping table updated

Annotations:

- Tokenizer renamed into Text Normalization:
 - a lot of debug mainly for English language, and punctuation managements
 - new option: can output a customized tier.
- Repetitions:
 - some debug

Communication:

- Add a description document of the orthographic transcription convention in the package.
- Web page updated, new tutorials available
- Documentation updated
- Better information about the licenses

8.5.9 SPPAS 1.9.1

(2017, 1st September)

Programming:

- Bug correction in the ELAN reader.
- Bug correction with quotation marks of the Praat writer.

Resources:

- Add an acoustic model for English, including laughter and noises.

8.5.10 SPPAS 1.9.2

(2017, 6th October)

Programming:

- Bug correction in the diagnosis of audio files.
- Package “anndata” continued, in the scope of replacing “annotationdata”.
- Acoustic model training procedure debugged and improved, code cleaned, scripts updated, etc.

8.5.11 SPPAS 1.9.3

(2017, 18th October)

Programming:

- Critical bug correction in Alignment: error when loading the tiedlist.
- Correction of TextNormalization of broken words: “-” is now not removed.

Resources:

- Italian pronunciation dictionary updated.
- French vocabulary updated: list of compound words revised.

Known bugs:

- Spanish alignment does not work: corrupted acoustic model

8.5.12 SPPAS 1.9.4

(2018, 15th January)

Programming:

- Script to train acoustic models updated
- Script to evaluate alignments updated
- Bug corrections of the search tier system

Resources:

- Spanish:
 - new pronunciation dictionary
 - new acoustic model

8.5.13 SPPAS 1.9.5

(2018, 26th April)

Programming:

- I/O development of the package “anndata”.
- Reading and writing of annotated files is now based on the new ‘anndata’ package instead of the old ‘annotationdata’ package. This was a big issue and it allows a better support of annotated files (less bugs and the amount of lost information is drastically reduced).

8.5.14 SPPAS 1.9.6

(2018, 25th May)

Resources:

- New: support of German language for Text Normalization (except num2letter), Phonetization and Alignment.
- French: new acoustic model with an updated phoneset and a better accuracy.
- Laugh and noise are uniformly represented in all languages.
- English: context-independent model is the default, instead of context-dependent model, for Alignment.

Development

- debug of the Transcriber file reader.
- debug in the management of the hierarchy in file readers/writers.
- Resources package: add compatibility with PLS (W3C xml) pronunciation dictionaries

Known bugs:

- Syllabification of French: the configuration file is not correct.

8.5.15 SPPAS 1.9.7

(2018, 23th July)

Development

- new filter system in `anndata`
- IPU's segmentation: debug in fixing automatically the threshold for the search of silences (can't be negative!)
- Annotations: improved messages for the procedure outcome report
- Annotations: Text normalization is based on `anndata` API instead of `annotationdata`
 - the result is now a sequence of individual tokens (each token is a `sppasLabel()`), instead of a single string separating tokens with space.
- Annotations: Phonetization is based on `anndata` API instead of `annotationdata`
 - the result is now a sequence of `sppasLabel()` with alternative tags: each tag corresponds to a pronunciation variant
- Annotations: Syllabification re-programmed and now based on `anndata` API
 - only sequences of phonemes are syllabified (i.e. no silence, no laugh nor noise in the result)
 - `O` class name changed into `P` class name
 - do not generate the tier “structures” tier anymore
 - generating the “classes” tier is optional
- Annotations: INTSINT is based on `anndata` API instead of `annotationdata`
- IPUScriber is based on `anndata` API instead of `annotationdata`

Resources:

- French syllabification rules updated to be compliant with the new phoneset (used since version 1.9.6)

Documentation

- updated chapter 6: scripting with Python and SPPAS. It is now based on `anndata` API instead of `annotationdata`.

8.5.16 SPPAS 1.9.8

(2018, 06th September)

e-mail contact is changed to:

- [*contact@sppas.org*](mailto:contact@sppas.org) for general questions of users
- [*develop@sppas.org*](mailto:develop@sppas.org) for developer questions or for a bug alert

Development

- bug correction for GMT with a negative value, like GMT-4.
- most of the scripts are based on `anndata` API instead of `annotationdata`
- dependency to markdown removed
- new solution to work with global variables with the new package `config` and the new classes like `sppasGlobalSettings`, `sppasPathSettings`, ...

- translation management moved in package `config`
- better way to work with imports
- increased pep8 and pep257 compatibility (should be continued)
- `anndata` is used in `DataRoamer`, `Visualizer`, `DataFilter`
- new script `trsshift.py` to shift the transcription of a delay in time

Documentation

The API documentation is based on Sphinx <http://www.sphinx-doc.org>.

8.5.17 SPPAS 1.9.9

(2018, 23th October)

Development

- package for the ‘alignment’ is fully re-structured (re-implemented partially), new unittests are added, and compatibility with python 3, pep8 and pep257.
- new GUI based on python3 + phoenix: it prints an information message in case SPPAS is launched with these versions.
- package ‘presenters’ updated: unittests, python 3, pep8 and pep257.

Plugins

- `sampa2ipa` included to the SPPAS package
- `audiosegmenter` included to the SPPAS package

Annotations

- IPU's Segmentation removed. It is replaced by:
 1. Search for IPU's: to find silences/tracks from an audio file
 2. Fill in IPU's: to find silences/tracks from an audio file and its transcription
 3. `AudioSegmenter` plugin: to segment audio files into several tracks
- `annotation.py`: bug correction with the default output extension
- bug correction when the phonetization was unknown

Resources

- add a file ‘monophones’ into each model, for the compatibility with `HVite`.
- errors of the acoustic models of “spa” and “nan” corrected.

8.5.18 SPPAS 2.0

(2019, 4th January)

The main change of this release is that the package `annotationdata` has been removed, so that all packages are using `anndata` instead. One of the most important consequence of this change is that all packages, except the `wxGUI`, are compatible with both Python 2.7 and Python 3.4+.

Annotations

- Alignment: bug correction if the last unit is empty.
- Self-Repetitions and Other-Repetitions split into two different packages
- Un-used package ‘Chunks’ removed
- Better management of annotations: `sppasBaseAnnot`, `sppasParam` and `sppasManager` revised.

UI

- All programs in `bin` folder updated. A ‘manual mode’ and an ‘auto mode’ are available in the annotations to fix input/output, and a `-log` option is proposed to save a report.
- Plugins: `marsatagplugin` added and debugged.

8.5.19 SPPAS 2.1

(2019, 28th February)

Development

- Praat TextGrid: it’s now allowed to use the symbol ‘=’ in annotation labels.
- Use of ‘json’ files instead of ‘ini’ files to configure SPPAS, the annotations and the plugins.
- New package ‘analysis’ with the filter and statistic systems.

Annotations

- Search for IPU: the program has been improved and evaluated.
- Alignment: priority is given to standard tokens (if EOT).
- Alignment: problem of whitespace in filenames solved.

UI

- Plugins: new plugin to remove un-translated IPUs and to re-index the IPUs
- extended internationalization of the messages (for English and French)

8.5.20 SPPAS 2.2

(2019, 09th May)

sppas.bat and sppas.command, the two main ways to launch the GUI, were fully re-written in order to search for 'pythonw' command first. It results in the following advantages: - it increases the compatibility with MacOS systems; - it allows to not display the dark frame of 'python' under Windows.

Development

- New package 'files' for a further use.
- Increased compatibility of file formats: can read alternative tags with {} system, whitespace vs CR, etc.

Plugins

- Debug of the plugin to clean IPU's.

Annotations

- Search for IPU's: a special threshold value is set when the audio recording has a very low median value (ie very low volume values)
- Normalization/Phonetization/Alignment/Syllabification can work without audio
- Search for IPU's and Fill in IPU's: problem with upper/lower extension solved

GUI

- "Delete" button of the "File Explorer": files are no longer definitively deleted. Instead, they are moved into an hidden folder with name '.trash' in the package of SPPAS.
- wxFrame to display log messages.
- Annotations in GUI: better compatibility to annotate written texts

Resources

- fra.dict: corrected 6 errors with a non-existing 'eu' sound

8.5.21 SPPAS 2.3

(2019, 25th June)

Development

- JSON configuration files of sppas modified.
- package "files" is used: the annotations manager is using a workspace instead of a list of files.

- the GUI based on wx4 is managing properly the workspaces and the annotations: a page with “Files” is displayed and allows to “add/remove/delete” files in the list, “import from/export to/pin&save/rename” workspaces, to “create/edit/delete” references and to “check with filters/check all/associate/dissociate” files and references.
- Workspaces are saved in “JSON” format in the “workspaces” folder.

Annotations

- New annotation “Activity”. This annotation was previously available as an option of “Alignment”.
- Text Normalization: the Num2Letter module has been fully re-programmed.
- Text Normalization: new option “occurrence & duration” to estimate the number of tokens of each IPU and the duration of this latter.
- Annotations are categorized as: STANDALONE, SPEAKER or INTERACTION. The GUI based on wx3 only shows the STANDALONE ones, but the GUI based on wx4 can deal with all of them.

Resources

- New folder “num” with dictionaries of numbers for cmn, eng, fra, ita, jpn, khm, pol, por, spa, vie.

8.5.22 SPPAS 2.4

(2019, 26th June)

Development

- the GUI based on wx4 is managing the plugins
- bug correction in ‘files’ package

Annotations

- New annotation “Re-Occurrences”, of type INTERACTION. Can only be used with the CLI or the GUI based on wx4.
- Either one or two new options in all the annotations:
 - inputpattern: to choose the pattern of the input file (like “-token” for Phonetization)
 - outputpattern: to choose the pattern of the output file (like “-token” for Tokenization)
- CLI: the script annotation.py is fully re-implemented

Resources

- New pronunciation dictionary for Iranian Persian (pes), for Phonetization
- New acoustic model for Iranian Persian (pes), for Alignment

8.5.23 SPPAS 2.5

(2019, 30th July)

Development

- the GUI based on wx4 can convert files.

Annotations

- New annotation “RMS” to estimate the Root-mean square of an audio file in given intervals

8.5.24 SPPAS 2.6

(2019, 1st October)

Various

- New script ‘dictsampa.py’ to convert a pronunciation dictionary in IPA into SAMPA encoding
- New script ‘clippingrate.py’ to estimate the clipping rate at several factors in sub-parts of the audio file
- New plugin ‘StatGroups’ to estimate distributional statistics on sequences of numbers in annotations.

Development

- the GUI based on wx4 can convert files (debugged and extended)
- the GUI based on wx4 can display the content of a file (ListView) in the page “Analyze”
- GUI: “Save all” button debugged in DataFilter (remaining bug in DataRoamer)

8.5.25 SPPAS 2.7

(2019, 2nd December)

Various

- Add the possibility to remove un-labelled annotations in tiers (in DataFilter)
- Plugin “StatGroups” updated: it can optionnally create a tier with the not-selected intervals

Resources

- Modified French lexicon: include words of Quebec French
- New Quebec French pronunciation dictionary
- New Quebec French acoustic model
- Modified French syllabification file: added vowels of Quebec French
- Tiedlist of Italien model re-introduced... Alignment of “ita” is once again completely functional

Development

- the GUI based on wx4: in “Files”, the list of files uses collapsible panels instead of a table

8.5.26 SPPAS 2.8

(2020, 17th January)

Two new scripts (sppas-py3.bat and sppas-py3.command) added to run directly the new GUI based on py3+wx4.

Development

- the GUI based on wx4: some debug, improved compatibility with the 3 OS and DataFilter and DataStats implemented in the page “Analyze”

Annotations

- New annotation: Stop Tags: estimate if tags of annotations are relevant or not. Relevance is estimated like for Other-Repetitions detections.

8.5.27 SPPAS 2.9

(2020, 03th April)

This is the last version for which both Python 2.7 and Python 3.5+ are supported. Running SPPAS with Python 2.7 will not be maintained past 2020, May when SPPAS 3.0 will be released.

8.5.28 Development

- the GUI based in wx4 is extended and debugged: audio roamer and ipuscriber integrated in the page ‘Analyze’, a multi-player is added - it can play several media at a time.

8.5.29 Resources

- updated dictionary and acoustic model of Polish language

8.5.30 Annotations

- New annotation LexMetric to estimate occurrences and rank

8.5.31 Known bugs:

In the page ‘Analyze’, the view “Multi-Player” does not properly displays the annotation boundaries and the scrollbar is not updated when needed.

8.6 Migrate to Python 3

Python 2.7 reached the end of its life. This version expects Python to 3.x as Python 2.7 is no longer maintained. No new bug reports, fixes, or changes will be made to SPPAS when used with Python 2.

A new setup program allows to install external programs to enable some features of SPPAS. It includes 'wxpython' to enable the Graphical User Interface.

8.6.1 SPPAS-3.0

(2020, 20th May)

Known bugs

- Does not support Julius 4.5+

GUI:

- The version based on wxPython 3 is deprecated. Instead, the one based on wxPython 4 is the default but some features are still not implemented.
- A large amount of debug of the GUI. Remaining bug: In the page 'Analyze', the view "Multi-Player" does not properly update the scrollbar when needed.
- Page annotate: can display the content of any report of the list, can delete existing reports.

Development:

- new package "preinstall" and new script preinstall.py to install external programs
- Package "files" renamed to "wkps". Updated classes and methods for a better support of reading and writing wjson files; a version 2.0 is then introduced. Add of a "MISSING" state allowing to properly manage missing entries of a workspace, relative paths added, etc.
- config package, classes and methods fully verified/modified including a new sppasAppConfig() class
- Modified **main** to enable to launch SPPAS GUI with the command: "python3 sppas"
- new bin "juliusdownload.py" to automatize the installation of julius under Windows.

Resources:

- Updated lexicon, dictionary and acoustic model for Naija Language.

Automatic annotations:

- Automatic alignment, when EOT, is rescuing a failed interval alignment with the "Tokens" tier instead of the default one.

8.6.2 SPPAS-3.1

(2020, 10th July)

Annotations:

- new annotation FaceDetection to detect faces in an image
- new annotation FaceLandmark to find coordinates of 68 points on an image representing a face, and only one
- new annotation SpkLexRep to find what are the tokens or phases repeated from one file to another one of the same speaker
- linguistic resources are downloaded and installed with the setup or directly from Ortolang web site

Others:

- renamed .deps~ into .app~

8.6.3 SPPAS-3.2

(2020, 4th September)

Known bugs

- Under Windows, when a ComboBox is clicked, the Log Window is focused.

Annotations

- better support of antx files
- new FaceTracking annotation to detect faces of a video, still under development, and not in the GUI.
- transfer the metadata of the input transcription to the output transcription. Reliable only for annotation formats that support metadata.
- face detection: sort detected faces by confidence score

Development

- bug correction and tests added
- debug of the features reading with UTF-8 encoding allowing the setup to work properly
- new videodata package to read and manipulate video files.

GUI

- The Home page includes links to the SPPAS web site
- some wx.ComboBox replaced by our sppasComboBox to look the same on all platforms
- Keyboard shortcuts added to message dialogs:
 - ESC: Cancel
 - Return: OK
 - n: No
 - y: Yes
- New keyboard shortcuts in the “Files” page of the main frame:

- alt + a: add files
- alt + e: export workspace
- alt + i: import workspace
- alt + s: pin&save workspace
- alt + n: rename the workspace
- alt + r: create a new reference
- alt + f: check some files
- alt + g: check all files
- alt + l: link checked files to checked references

8.6.4 SPPAS-3.3

(2020, 20th October)

Package

- “.logs” and “.trash” folders are no longer hidden. Now, they are named “logs” and “trash”.

GUI

- The page Analyze is simplified: there’s no longer a notebook to open files into tabs, and there’s no longer different “views”. Instead, 1/ to edit the text content of a file, the page “Files” has a new button “Edit checked” in the top toolbar; and 2/ to view files in a time-line style, a new page “Editor” was created. The latter is still under development.

Annotations

- Support of versions 4.5+ of the Julius CSR Engine, used by the ”Alignment automatic annotation.
- new FaceTrack annotation to detect faces into a video. Currently not fully tested.

Development

- new package “videodata” to support video files with opencv.
- preinstall: a different way to search for the python command. It should solve problems when the Python3 executable is “python” command instead of “python3”.

8.6.5 SPPAS-3.4

(2020, 4th December)

GUI

- web links to the reference publications are indicated for each automatic annotations, either in PDF or URL.
- the main app is no longer using the option “useBestVisual” - a user reported that the app crashes with anaconda/ubuntu if enabled.

- some debug and improvement of the page Analyze. Now, it looks better under Windows.
- the page Editor is continued. It allows to:
 1. open audio, video and transcription files (no limit on the number of files);
 2. play all audio and video files really synchronously: the max delay between media players is 10ms;
 3. a slider allows to select 5 different periods to play: either the whole duration, or the visible part, or the selected part, or the visible part until the selected part or the visible part after the selected part.
 4. scroll/zoom the files to change the visible part;
 5. display annotations both in a spreadsheet and in a timeline;
 6. select an annotation;
 7. edit the annotation content: a text entry allows to edit the labels of an annotation either in a serialized string, in XML or in JSON.
 8. add/delete/split annotations.
 9. save.

Development

- setup.bat was modified, expecting it can install wxpython even if python was not installed from the Windows store. not tested.
- analyze: added a new option in the filter system when using relations. It allows to crop annotations of X when they are overlapping Y (overlaps, overlapped by and contains).

8.6.6 SPPAS-3.5

(2021, 20th January)

GUI

- Page Annotate: the installation of new languages and new annotations is enabled.
- Page Analyze:
 1. view of the waveform of audio files
 2. the files of the timeline can be sorted
 3. in the timeline, right-click on a transcription file to check-uncheck tiers

Annotations

- LexMetric: bug correction of the supported files. No remaining unjustified error message.
- FaceDetection: can detect faces on both an image or a video.
- FaceMark renamed FaceSights: can fix 68 sights of detected faces on both an image or a video.

Development

- two new scripts: 1/ to convert a folder of images into a video and 2/ to export a video into a folder of images

8.6.7 SPPAS-3.6

(2021, 23th February)

Plugins

- cleanipus updated: solved import error and version updated to 1.1

Annotations

- INFO1220 migrated from report to logging
- replaced by dummy for unknown phonetizations
- FaceDetection: more options, more models
- New annotation FaceIdentity to assign an identity on detected faces

Development

- bug correction in audioframes with an exception not properly raised
- anndata: sppasWEKA renamed to sppasTable and enhanced with export of TRA files – Table Rich Annotations
- installer: added numpy to video feature with version > 1.20 due to the Windows 2004 update and its consequences...
- workspaces: when reading wjson, the references are properly assigned to files
- sppasLPC renamed to sppasCuedSpeech

CLI

- the script trstotable.py is replacing trstoweka.py
- other-repetition script can take a workspace as input

8.6.8 SPPAS-3.7

(2021, 14th April)

Annotations

- creating a merged file is enabled by default
- Overlaps: new INTERACTION annotation to create a tier with overlapping activities.
- Alignment: a better support of non us-ascii characters
- Stopwords: no remaining error about file extension.
- FaceDetection: the default is to load both a DNN and a HAAR model instead of two DNNs. Enlarged portrait size (if option is enabled).
- FaceIdent: can export a smoothed portrait video.
- CuedSpeech: can export the video with the position of the 5 vowels and a number to represent the consonant during the audio key. The resulting tier with the key is made of two labels: the consonant number and the vowel number.

- OtherRepetitions: 3 more tiers about the OR source are created (word strain, nb of labels, source type). Bug correction in the position of the first found echo.
- SelfRepetitions: 3 more tiers about the SR source are created (word strain, nb of labels, source type)
- Linguistic resources and annotation resources are no longer downloaded from ortolang but from gandi instead. Manual download is still on ortolang (when it works!)

Analysis

- filter system: New filter 'nlab' on the number of labels of annotations

Development

- bug correction in video coords reader
- videodata: the FPS is not converted to integer anymore, now it can be a float. Support of .mov files with H264 codec.
- updated sppasui.json to re-rank annotations of faces
- images: extended support with transparency and overlay added

GUI

- page files: bug correction when adding a folder, automatically adjust the column width of the filename when adding new files
- page analyze: new filter added to the SingleFilter, modified files can be saved
- page editor: annotation list view and labels edit re-organized, a new rise panel for the timeline view of file allows to see action buttons when collapsed instead of when expanded. No remaining infinite loop (macos, linux) when displaying tiers. A proportional frame when reading a video.
- Problem with foreground color of lists solved (macos, linux)

8.6.9 SPPAS-3.8

(2021, 26th May)

Annotations

- New annotation IVA - Interval Values Analysis.
- Lexical Metric: Add the number of annotations and the number of labels in segments. Segments are fixed from a list of separators.
- Fill in IPU: more options to fix shift start and shift end.
- CuedSpeech: bug correction in key generation at the end of an IPU with a consonant.
- CuedSpeech: return 2 different tiers with the key, either with a single combined label or with two different labels (one for C then one for V).
- Alignment: a better way to convert point to intervals in case of alignment without audio.
- Other Repetitions: optionally can add a tier with all echo candidates.
- Extended search for syllables tier name: it allows to apply TGA to any tier with 'Syll' in its name.

GUI

- buttons have new custom events: it allows to customize their look when mouse is entering, leaving, ...
- page editor: annotation boundaries can be moved. The selected bound is in green if several annotations share the same time value, or in red if the bound corresponds to only one time value. Both the midpoint and the radius can be modified.
- page editor: can create an annotation in the timeline by dragging with the mouse.
- page editor: Some minor changes in the look of the page.
- page editor: bug correction with threads in video player under MACOS. Allows to play several videos at a time, like on the other systems.
- page editor: bug correction in the list of annotations when adding before.
- page files: bug correction when adding a non-valid file.
- page files: adding the parent folder '..' is forbidden.

Development

- preinstall: added linux+microsoft in the list of supported distrib for those who installed Ubuntu from the Windows Store
- a global `sppasLogSetup()` instance is declared in the config.
- the script `slmtrain.py` can take sentences from stdin and then write the probas on stdout.
- anndata: refactor of `sppasRW` into `sppasTrsRW`.
- anndata: added a `trs_type` in all reader/writer classes and in `sppasTrsRW` file properties. Allows to classify annotated files into: ANNOT, MEASURE or TABLE.
- annotations: by default, automatic annotations can only create files of the type ANNOT (except if overwrite the method).
- config: add a log error message if no write access to `sppas` directory.

Scripts

audioinfo: new options to get clipping values in details or no clipping values at all

8.6.10 SPPAS-3.9

(2021, 6th July)

Annotations

- More logging messages for a better understanding of the automatic annotations procedures
- bugs corrections in RMS and SpkLexRep
- The “merge” can now be checked/unchecked in the page to annotate. If checked, it merges only checked files of each root.

GUI

- Under MacOS, the GUI application is not crashing anymore on exit - previously it was randomly either not crashing, or bus error or segmentation fault
- Make the GUI work correctly with the MacOS Dock

- Page File: Edit files button is moved to the vertical toolbar and two buttons are added to remove missing files and to add all files sharing checked roots
- Page Editor: An elaborated search dialog is available in the editor in order to find label patterns in annotations of one or several tiers

Development

- setup for Windows is simplified
- a new class `utils.SppasFiles()` allows a better control of the files to manage in the automatic annotations
- annotations: No longer required vs optional inputs. A more generic solution is used instead in all the annotations

8.6.11 Next...

The next release is expected on 15th September. It will be the end of the python 2.7 support. Priority is also given to the page Editor. The expected changes are:

- an easier way to explore the timeline – including scrolling with the mouse;
- a nicer list of annotations – the current list is very useful but ugly;
- a one-by-one image player of videos.

Appendix

9.1 List of error messages

SPPAS introduced a system for the internationalization of its messages. In the same time, a quality and a number was assigned progressively to each of them in the packages. This system is aimed at providing the users with appropriate error messages and a deeper understanding of the problems.

9.1.1 Global errors

:ERROR 0001:

This is an un-classified error, with the generic name “sppasError”. A message after the error should indicate the problem.

:ERROR 0513:

This is the `sppasPermissionError`, with the message: access to {place} is denied. It occurs when SPPAS was not able to create a file or a directory in some place of the computer. The most frequent cause is that the user do not have permissions to write in the directory where the SPPAS package is installed, or the rights are configured incorrectly or not configured at all.

To work around this issue, use one of the following workarounds as appropriate for your situation:

1. install SPPAS package in a place you have appropriate rights;
2. ask your OS Administrator to modify the rights of the directory in which SPPAS package is installed;
3. launch SPPAS with Administrator rights (not recommended).

9.1.2 From “annotations” package

:ERROR 1010:

Unknown option with key {key}.

:ERROR 1020:

This error has the message: Empty input tier {name}. It occurs when an automatic annotation is expecting an annotated tier but the given one does not contain annotations.

:ERROR 1030:

Missing input tier. Please read the documentation.

:ERROR 1040:

Bad input tier type.

:ERROR 1050:

Inconsistency between the number of intervals of the input tiers. Got: { :d } and { :d }.

9.1.3 From “utils” package

:ERROR 1210: The directory {dirname} does not exist.

:ERROR 1220: The directory {dirname} does not contain relevant data.

9.1.4 From “audiodata” package

:ERROR 2000: No audio file is defined.

:ERROR 2005: Audio type error: not supported file format {extension}.

:ERROR 2010: Opening, reading or writing error.

:ERROR 2015: No data or corrupted data in the audio file {filename}.

:ERROR 2020: {number} is not a right index of channel.

:ERROR 2025: From {value1} to {value2} is not a proper interval.

:ERROR 2050: No channel defined.

:ERROR 2060: Channels have not the same sample width.

:ERROR 2061: Channels have not the same frame rate.

:ERROR 2062: Channels have not the same number of frames.

:ERROR 2070: Invalid sample width {value}.

:ERROR 2080: Invalid frame rate {value}.

9.1.5 From “calculus” package

:ERROR 3010: Both vectors p and q must have the same length and must contain probabilities.
:ERROR 3015: Value must range between 0 and 1. Got { :f}.
:ERROR 3016: Probabilities must sum to 1. Got { :f}.
:ERROR 3025: Error while estimating Euclidian distances of rows and columns.
:ERROR 3030: The given data must be defined or must not be empty.
:ERROR 3040: Value { value } is out of range: expected value in range [{ min_value }, { max_value }].

9.1.6 From “plugins” package

:ERROR 4010: Missing plugin configuration file.
:ERROR 4014: Missing section { section_name } in the configuration file.
:ERROR 4016: Missing option { :s } in section { :s } of the configuration file.
:ERROR 4020: Unsupported plugin file type.
:ERROR 4024: Unsupported plugin file type.
:ERROR 4030: A plugin with the same name is already existing in the plugins folder.
:ERROR 4040: No plugin with identifier { plugin_id } is available.
:ERROR 4050: No such plugin folder: { :s }.
:ERROR 4060: A plugin with the same key is already existing or plugin already loaded.
:ERROR 4070: { command_name } is not a valid command on your operating system.
:ERROR 4075: No command was defined for the system: { :s }. Supported systems of this plugin are: { :s }. ”””
:ERROR 4080: No option with key { :s }.

9.1.7 From “resources” package

:ERROR 5005:

Encoding error while trying to read the file: { name }.

:ERROR 5010:

Error while trying to open and read the file: { name }.

:ERROR 5015:

Read file failed at line number { number }: { string }.

:ERROR 5020:

The n value of n-grams pattern matching must range [1; { maximum }]. Got { observed }.

Others:

:ERROR 5022: The gap value of pattern matching must range [0;{maximum}]. Got {observed}.

:ERROR 5024: The score value of unigrams pattern matching must range [0;1]. Got {observed}.

:ERROR 5030: The dump file can't have the same extension as the ASCII file ({extension}).

:ERROR 5040: The count value must be positive. Got ({count}).

9.1.8 From “structs” package

:ERROR 6010: {meta} is not a known meta information.

:ERROR 6020: Unknown resource type: expected file or directory. Got: {string}.

:ERROR 6024: The resource folder {dirname} does not exists.

:ERROR 6028: The language must be “und” or one of the language list. Unknown language {lang}.

9.1.9 From “models” package

:ERROR 7010: Expected a {data_name} of type {expected_type}. Got {data_type} instead.

:ERROR 7500: The file {!s:s} contains non UTF-8 characters: {!s:s}.

:ERROR 7505: Fail formats: unrecognized file format {!s:s}.

:ERROR 7510: Fail formats: the folder {!s:s} does not contain a known model.

:ERROR 7515: No model found or empty model in {!s:s}.

9.2 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software

does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not

have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket

the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.