HAL
open science

SPPAS - the automatic annotation and analysis of speech (version 1.9.3)

Brigitte Bigi

To cite this version:

Brigitte Bigi. SPPAS - the automatic annotation and analysis of speech (version 1.9.3) . 2018. hal-01414600v2

HAL Id: hal-01414600
https://hal.science/hal-01414600v2

Preprint submitted on 9 Jan 2018 (v2), last revised 21 Nov 2022 (v8)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

SPPAS

the automatic annotation and analysis of speech

(version 1.9.1)

by Brigitte Bigi

Any and all constructive comments are welcome

Copyright © 2011-2017 - 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".

Contents

1 Introduction **1**

1.1 What is SPPAS? . . . 1

1.1.1 Overview . . . 1

1.1.2 Licenses . . . 2

1.1.3 User engagement . . . 2

1.1.4 Need help . . . 2

1.1.5 Supports . . . 3

1.1.6 Author . . . 3

1.2 Getting and installing . . . 4

1.2.1 Websites . . . 4

1.2.2 External programs . . . 4

1.2.3 Download and install SPPAS . . . 4

1.2.4 The SPPAS package . . . 5

1.2.5 Update . . . 6

1.3 Capabilities . . . 6

1.3.1 What SPPAS can do? . . . 6

1.3.2 How to use SPPAS? . . . 7

1.4 Main and important recommendations . . . 7

1.4.1 About files . . . 7

1.4.2 About automatic annotations . . . 8

1.4.3 About linguistic resources . . . 8

1.5 About this documentation . . . 10

2 Interface **11**

2.1 Introduction . . . 11

2.2 GUI Main frame . . . 11

2.2.1 Launch SPPAS . . . 11

2.2.2 The tips . . . 12
2.2.3 The menu . . . 12
2.2.4 The file explorer . . . 13
2.2.5 Settings . . . 15
2.2.6 About . . . 16
2.2.7 Help . . . 16
2.2.8 Plugins . . . 16
2.3 CLI . . . 18

3 Automatic Annotations **21**
3.1 Introduction . . . 21
3.1.1 About this chapter . . . 21
3.1.2 Annotations methodology . . . 21
3.1.3 File formats and tier names . . . 23
3.1.4 Recorded speech . . . 23
3.1.5 Automatic Annotations with GUI . . . 23
3.1.6 Automatic Annotations with CLI . . . 24
3.1.7 The procedure outcome report . . . 26
3.2 New language support . . . 30
3.3 Orthographic Transcription . . . 30
3.4 Inter-Pausal Units (IPUs) segmentation . . . 31
3.4.1 Overview . . . 31
3.4.2 Silence/Speech segmentation . . . 31
3.4.3 Silence/Speech segmentation time-aligned with a transcription . . . 33
3.4.4 Split into multiple files . . . 35
3.4.5 Perform IPUs Segmentation with the GUI . . . 35
3.4.6 Perform IPUs Segmentation with the CLI . . . 35
3.5 Text normalization . . . 36
3.5.1 Overview . . . 36
3.5.2 Adapt Text normalization . . . 37
3.5.3 Support of a new language . . . 37
3.5.4 Perform Text Normalization with the GUI . . . 37
3.5.5 Perform Text Normalization with the CLI . . . 38
3.6 Phonetization . . . 39
3.6.1 Overview . . . 39

3.6.2 Adapt Phonetization . . . 39
3.6.3 Support of a new language . . . 40
3.6.4 Perform Phonetization with the GUI . . . 40
3.6.5 Perform Phonetization with the CLI . . . 40
3.7 Alignment . . . 42
3.7.1 Overview . . . 42
3.7.2 Adapt Alignment . . . 42
3.7.3 Support of a new language . . . 42
3.7.4 Perform Alignment with the GUI . . . 43
3.7.5 Perform Alignment with the CLI . . . 44
3.8 Syllabification . . . 44
3.8.1 Overview . . . 44
3.8.2 Adapt Syllabification . . . 45
3.8.3 Support of a new language . . . 46
3.8.4 Perform Syllabification with the GUI . . . 46
3.8.5 Perform Syllabification with the CLI . . . 46
3.9 Repetitions . . . 47
3.9.1 Overview . . . 47
3.9.2 Adapt Repetitions and support of a new language . . . 47
3.9.3 Perform Repetitions with the GUI . . . 47
3.9.4 Perform Repetitions with the CLI . . . 48
3.10 Momel and INTSINT . . . 48
3.10.1 Momel (modelling melody) . . . 48
3.10.2 Encoding of F0 target points using the “INTSINT” system . . . 49
3.10.3 Perform Momel and INTSINT with the GUI . . . 50
3.10.4 Perform Momel and INTSINT with the CLI . . . 50

4 Resources for Automatic Annotations 51
4.1 Overview . . . 51
4.2 French . . . 52
4.2.1 List of phonemes . . . 52
4.2.2 Pronunciation dictionary . . . 54
4.2.3 Acoustic Model . . . 54
4.2.4 Syllabification configuration file . . . 54
4.3 Italian . . . 54

4.3.1 List of phonemes . . . 54
4.3.2 Pronunciation dictionary . . . 56
4.3.3 Acoustic Model . . . 57
4.3.4 Syllabification configuration file . . . 57
4.4 Spanish . . . 57
4.4.1 List of phonemes . . . 57
4.4.2 Pronunciation Dictionary . . . 59
4.4.3 Acoustic Model . . . 59
4.5 Catalan . . . 59
4.5.1 List of phonemes . . . 60
4.5.2 Pronunciation dictionary . . . 62
4.5.3 Acoustic Model . . . 62
4.6 English . . . 62
4.6.1 List of phonemes . . . 62
4.6.2 Pronunciation dictionary . . . 65
4.6.3 Acoustic Model . . . 65
4.7 Polish . . . 65
4.7.1 List of phonemes . . . 65
4.7.2 Pronunciation Dictionary . . . 67
4.7.3 Acoustic Model . . . 68
4.8 Portuguese . . . 68
4.8.1 Pronunciation Dictionary . . . 70
4.8.2 Acoustic Model . . . 70
4.9 Mandarin Chinese . . . 70
4.9.1 List of phonemes . . . 70
4.9.2 Pronunciation dictionary . . . 72
4.9.3 Acoustic model . . . 73
4.10 Southern Min (or Min Nan) . . . 73
4.10.1 List of phonemes . . . 73
4.10.2 Pronunciation Dictionary . . . 75
4.10.3 Acoustic Model . . . 76
4.11 Cantonese . . . 76
4.11.1 List of phonemes . . . 76
4.11.2 Acoustic Model . . . 78
4.12 Japanese . . . 78

4.13 Korean . . . 79
4.13.1 List of phonemes . . . 79
4.13.2 Pronunciation dictionary . . . 81
4.13.3 Acoustic Model . . . 81
4.14 Naija . . . 81
4.14.1 List of phonemes . . . 81
4.14.2 Pronunciation Dictionary . . . 84
4.14.3 Acoustic Model . . . 84

5 Analyses **85**
5.1 Introduction . . . 85
5.2 DataRoamer . . . 87
5.3 AudioRoamer . . . 88
5.3.1 Properties of the audio file . . . 88
5.3.2 Playing audio files . . . 89
5.3.3 Want more? . . . 90
5.4 IPUscriber . . . 91
5.5 Visualizer . . . 93
5.6 DataFilter . . . 94
5.6.1 Filtering annotations of a tier: SingleFilter . . . 94
5.6.2 Filtering on time-relations between two tiers . . . 97
5.7 Statistics . . . 99
5.7.1 Descriptive statistics . . . 99
5.7.2 TGA - Time Group Analyzer . . . 100
5.7.3 User agreement . . . 101

6 Scripting with Python and SPPAS **103**
6.1 Introduction . . . 103
6.2 A gentle introduction to programming . . . 103
6.2.1 Introduction . . . 104
6.2.2 Variables: Assignment and Typing . . . 104
6.2.3 Basic Operators . . . 105
6.2.4 Data types . . . 105
6.2.5 Conditions . . . 106
6.2.6 Loops . . . 107
6.2.7 Dictionaries . . . 108

6.3 Scripting with Python . . . 108
6.3.1 Comments and documentation . . . 109
6.3.2 Getting started with scripting in Python . . . 109
6.3.3 Functions . . . 110
6.3.4 Reading/Writing files . . . 112
6.3.5 Python tutorials . . . 114
6.3.6 Exercises to practice . . . 114
6.4 The API of SPPAS to manage data . . . 114
6.4.1 Overview . . . 114
6.4.2 Why developing a new API? . . . 115
6.4.3 The API class diagram . . . 115
6.5 Creating scripts with annotationdata API . . . 117
6.5.1 Preparing the data . . . 117
6.5.2 Read/Write annotated files . . . 118
6.5.3 Manipulating a Transcription object . . . 119
6.5.4 Manipulating a Tier object . . . 119
6.5.5 Main information on Annotation/Location/Label objects . . . 120
6.5.6 Exercises . . . 121
6.5.7 Search in annotations: Filters . . . 121
6.5.8 Exercises . . . 123
6.6 More with SPPAS API . . . 123

7 References **125**
7.1 References about SPPAS . . . 125
7.1.1 How to cite SPPAS? . . . 125
7.1.2 SPPAS software description . . . 125
7.1.3 About Text Normalization . . . 126
7.1.4 About Phonetization . . . 126
7.1.5 About Forced-Alignment . . . 126
7.1.6 About Syllabification . . . 126
7.1.7 About Repetitions . . . 127
7.1.8 About analyses tools . . . 127
7.1.9 About the API . . . 127
7.1.10 Related references . . . 127
7.2 SPPAS in research projects . . . 129

7.2.1 MULTIPHONIA . . . 129
7.2.2 Amennpro . . . 129
7.2.3 Evalita 2011: Italian phonetization and alignment . . . 129
7.2.4 Cofee: Conversational Feedback . . . 130
7.2.5 Variamu: Variations in Action: a MUltilingual approach . . . 130

8 SPPAS Release notes **131**
8.1 The early versions . . . 131
8.1.1 Version 1.0 (2011, 9th March) . . . 131
8.1.2 Version 1.1 (2011, 7th June) . . . 131
8.1.3 Version 1.2 (2011, 23th July) . . . 131
8.1.4 Version 1.3 (2011, 12th December) . . . 131
8.2 The birth of SPPAS . . . 132
8.2.1 SPPAS 1.4.0 (2012, 14th June) . . . 132
8.2.2 SPPAS 1.4.1 (2012, 13th July) . . . 133
8.2.3 SPPAS 1.4.2 (2012, 2nd August) . . . 134
8.2.4 SPPAS 1.4.3 (2012, 10th October) . . . 134
8.2.5 SPPAS 1.4.4 (2012, 6th December) . . . 135
8.2.6 SPPAS 1.4.5 (2013, 15th January) . . . 135
8.2.7 SPPAS 1.4.6 (2013, 12th February) . . . 136
8.2.8 SPPAS 1.4.7 (2013, 25th March) . . . 136
8.2.9 SPPAS 1.4.8 (2013, 30th May) . . . 136
8.2.10 SPPAS 1.4.9 (2013, 3rd July) . . . 137
8.2.11 SPPAS 1.5.0 (2013, 2nd August) . . . 137
8.2.12 SPPAS 1.5.1 (2013, 29th August) . . . 138
8.2.13 SPPAS 1.5.2 (2013, 27th September) . . . 138
8.2.14 SPPAS 1.5.3 (2013, 25th October) . . . 139
8.2.15 SPPAS 1.5.4 (2013, 3rd December) . . . 139
8.2.16 SPPAS 1.5.5 (2013, 23th December) . . . 139
8.2.17 SPPAS 1.5.6 (2014, 28th January) . . . 140
8.2.18 SPPAS 1.5.7 (2014, 18th February) . . . 140
8.2.19 SPPAS 1.5.8 (2014, 18th March) . . . 140
8.2.20 SPPAS 1.5.9 (2014, 15th April) . . . 140
8.2.21 SPPAS 1.6.0 (2014, 22th May) . . . 141
8.2.22 SPPAS 1.6.1 (2014, 26th September) . . . 141

8.2.23 SPPAS 1.6.2 (2014, 21th October) . . . 142
8.2.24 SPPAS 1.6.3 (2014, 2nd November) . . . 142
8.3 The childhood of SPPAS . . . 142
8.3.1 SPPAS 1.6.4 (2014, 5th December) . . . 142
8.3.2 SPPAS 1.6.5 (2014, 17th December) . . . 143
8.3.3 SPPAS 1.6.6 (2015, 19th January) . . . 143
8.3.4 SPPAS-1.6.7 (2015, 16th February) . . . 144
8.3.5 SPPAS-1.6.8 (2015, 9th April) . . . 145
8.4 The development phase . . . 146
8.4.1 SPPAS-1.6.9 (2015, 14th May) . . . 146
8.4.2 SPPAS-1.7.0 (2015, 3th July) . . . 146
8.4.3 SPPAS-1.7.1 (2015, 5th August) . . . 147
8.4.4 SPPAS-1.7.2 (2015, 3th September) . . . 148
8.4.5 SPPAS-1.7.3 (2015, 9th October) . . . 148
8.4.6 SPPAS-1.7.4 (2015, 6th November) . . . 148
8.4.7 SPPAS-1.7.5 (2015, 11th December) . . . 149
8.4.8 SPPAS-1.7.6 (2016, 28th January) . . . 149
8.4.9 SPPAS-1.7.7 (2016, 30th March) . . . 150
8.4.10 SPPAS-1.7.8 (2016, 5th May) . . . 150
8.4.11 SPPAS-1.7.9 (2016, 3th June) . . . 151
8.5 The stabilization phase . . . 151
8.5.1 SPPAS-1.8.0 (2016, 30th August) . . . 151
8.5.2 SPPAS-1.8.1 (2016, 28th November) . . . 152
8.5.3 SPPAS-1.8.2 (2017, 18th January) . . . 152
8.5.4 SPPAS-1.8.3 (2017, 10th March) . . . 152
8.5.5 SPPAS 1.8.4 (2017, 10th April) . . . 153
8.5.6 SPPAS 1.8.5 (2017, 20th April) . . . 153
8.5.7 SPPAS 1.8.6 (2017, 19th June) . . . 153
8.5.8 SPPAS 1.9.0 (2017, 28th July) . . . 153
8.5.9 SPPAS 1.9.1 (2017, 1st September) . . . 154
8.5.10 SPPAS 1.9.2 (2017, 6th October) . . . 155
8.5.11 SPPAS 1.9.3 (2017, 18th October) . . . 155

9 Appendix **157**
9.1 List of error messages . . . 157
9.2 GNU Free Documentation License . . . 159

1

Introduction

1.1 What is SPPAS?

1.1.1 Overview

SPPAS - the automatic annotation and analyses of speech is a scientific computer software package written and maintained by Brigitte Bigi of the Laboratoire Parole et Langage. 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 can then automatize annotations and allows users to save time. In order to be used efficiently, SPPAS expects a rigorous methodology to collect data and to prepare them. Automatic annotations of SPPAS can be used either with a Command-line User Interface or a Graphical User Interface. So, there's no specific difficulty by using this software. Advanced users can also access directly the Application Programming Interface - API.

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.

Contact the author by e-mail to contribute to SPPAS:

- to improve the quality of the linguistic resources,
- to add new linguistic resources,
- to help in development,
- to add new annotations or analysis methods,
- or anything else!

1.1.2 Licenses

SPPAS software, except documentation and resources, are distributed under the terms of the GNU GENERAL PUBLIC LICENSE v3.

Documentation is governed by GNU Free Documentation License, version 1.3.

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".

A copy of both licenses is available in the package. 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.

1.1.3 User engagement

By using SPPAS, **you agree to cite the reference in your publications. See the chapter 7 of this documentation.

1.1.4 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, and it can also be explored with the Graphical User Interface by clicking on the `Help` button.

3. There is a F.A.Q. in the SPPAS web site. Topics can range from elementary "how do I" queries to advanced issues in scriptwriting. There's (or there will be) something there for everybody.

4. If none of the above helps, you may contact the author by e-mail. Another option is that you may send e-mail to the author. In both cases, it is very important to indicate clearly:

    1. your operating system and its version,
    2. the version of SPPAS (supposed to be the last one), and
    3. for automatic annotations, send the log file, and a sample of the data on which a problem occurs.

1.1.5 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

The introduction of Naija language is supported by the French National Research Agency (ANR) NaijaSynCor.

1.1.6 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, Aix-en-Provence, France.

Send an e-mail to the author: brigitte.bigi@gmail.com

Know more about the author: http://www.lpl-aix.fr/~bigi

Contributors

Here is the list of other contributors in programming:

- April 2012-June 2012: Alexandre Ranson;
- April 2012-July 2012: Cazembé Henry;
- April 2012-June 2013: Bastien Herbaut;
- March 2013-March 2014: Tatsuya Watanabe;
- April 2015-June 2015: Nicolas Chazeau;
- April 2015-June 2015: Jibril Saffi.

1.2 Getting and installing

1.2.1 Websites

Since January 2016, the main website of SPPAS is located at the following URL:

http://www.sppas.org

The source code with recent stable releases is hosted on github. From this website, anyone can download the current version and the development version. After signed-in, anyone can send comments and/or declare an issue:

https://github.com/brigittebigi/sppas

1.2.2 External programs

On the main website, it is possible to find information about the software requirements. In fact, other programs are required for SPPAS to operate. Of course, they must be installed before using SPPAS, and *only once*. This operation takes 5 up to 10 minutes depending on the operating system.

The following software are required:

1. Python 2.7.x
2. wxPython >= 3.0
3. julius >= 4.1

It is very (very very) important to take care about the version of Python. An installation guide is available on the website, depending on the operating system. **Please, closely follow the instructions.** Notice that administrator rights are required to perform the installations. In case of difficulty arising from the installation of such software, we invite you to consult the web first. It probably will provide the right solution. If, however, the problems were to persist, contact the author by e-mail.

1.2.3 Download and install SPPAS

The main website contains the `Download` page to download recent versions. There is a unique version of SPPAS which does not depend on the operating system. The only obvious difference depending on the system is how it looks on the computer screen.

Figure 1.1: Operating systems

SPPAS is ready to run, so it does not need elaborate installation, except for the other software required for SPPAS to work properly. 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 name of the path*.

The SPPAS package is compressed and zipped, so you will need to *decompress and unpack* it once you've got it.

1.2.4 The SPPAS 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 any user to adapt automatic annotations to its own needs. The package of SPPAS is then a directory with content as files and folders.

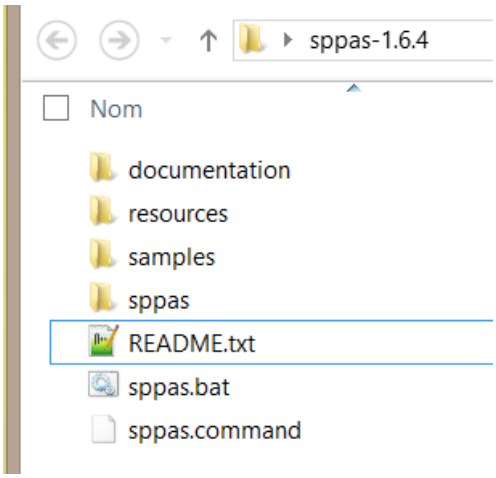


Figure 1.2: SPPAS Package content

The SPPAS package contains:

- the `README.txt` file, which aims to be read
- the files `sppas.bat` and `sppas.command` to execute the Graphical User Interface
- the `resources` directory contains data that are used by automatic annotations (lexicons, dictionaries, …)

- the `samples` directory contains folders of data of various languages, they are sets of annotations freely distributed to test features of SPPAS
- the `sppas` directory contains the program itself
- the `documentation` directory contains:
  - the copyright
  - the printable documentation
  - the printable version of the main reference published in "the Phonetician" journal
  - the folder `scripting_solutions` is a set of python scripts corresponding to the exercises proposed in the chapter "Scripting with Python and SPPAS"

1.2.5 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 functions and corrections.

Updating SPPAS is very easy and fast:

1. Download the last updated package from the SPPAS main web site;
2. Unpack the downloaded package;
3. Optionally, put the old one into the Trash.

1.3 Capabilities

1.3.1 What SPPAS can do?

Here is the list of features available to annotate automatically speech data and to analyze annotated files:

1. Automatic and semi-automatic annotations

   - *Momel*: modelling melody
   - *INTSINT*: Intonation

   - *IPUs segmentation*: speech/silence segmentation
   - *Text normalization*: tokenization, remove punctuation, etc
   - *Phonetization*: grapheme to phoneme conversion
   - *Alignment*: phonetic segmentation
   - *Syllabification*: group phonemes into syllables
   - *Repetitions*: detect self-repetitions, and other-repetitions

2. Analysis

   - *IPUscriber*: Manual orthographic transcription
   - *AudioRoamer*: Play, show information and manage speech audio files
   - *Statistics*: Estimates/Save statistics on annotated files
   - *DataRoamer*: Manipulate annotated files

- *DataFilter*: Extract data from annotated files (i.e. searching in data)
- *Visualizer*: Display sound and annotated files in a time-line

3. Plugins

   - *SAMPA to IPA*: convert SAMPA into IPA phonemes encoding
   - *SAMPA to Praat-IPA*: convert SAMPA into Praat phonemes encoding
   - *Classify phones*: create tiers with the phonemes classification (articulatory information)
   - *Marsatag*: apply MarsaTag French POS-tagger on time-aligned files
   - *sox*: call the Swiss Army Knife of sound processing utilities from SPPAS

1.3.2 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.

1.4 Main and important recommendations

1.4.1 About files

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

1. Speech audio files:

   - only `wav` and `au` files
   - only mono (= one channel)
   - frame rates: 16000hz
   - bit rate: 16 bits
   - good recording quality is expected. It is obviously required to never convert from a compressed file, as mp3 for example.

2. Annotated data files:

   - *UTF-8* encoding only
   - it is recommended to use only US-ASCII characters in file names (including the path)

Other frame rates and bit rates are accepted: the audio file is converted in 16 bits/16000Hz in a copy of the file then SPPAS is using the latter.

1.4.2 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 SPPAS package. The quality of the automatic annotations is largely influenced by **the quality of the linguistic resources**. The users can contribute to improve them.

1.4.3 About linguistic resources

Users are of crucial importance for resource development. They can contribute to improve them, and the author releases the improvements to the public, so that the whole community benefits. ## Interoperability and compatibility

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 convert files. 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.

SPPAS makes use of its internal data representation to convert files. A conversion then consists of two steps: First, the incoming file is loaded and mapped to the SPPAS data framework; and second, 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 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.

To summarize, SPPAS supports the following software with their file extensions:

- Praat: TextGrid, PitchTier, IntensityTier
- Elan: eaf

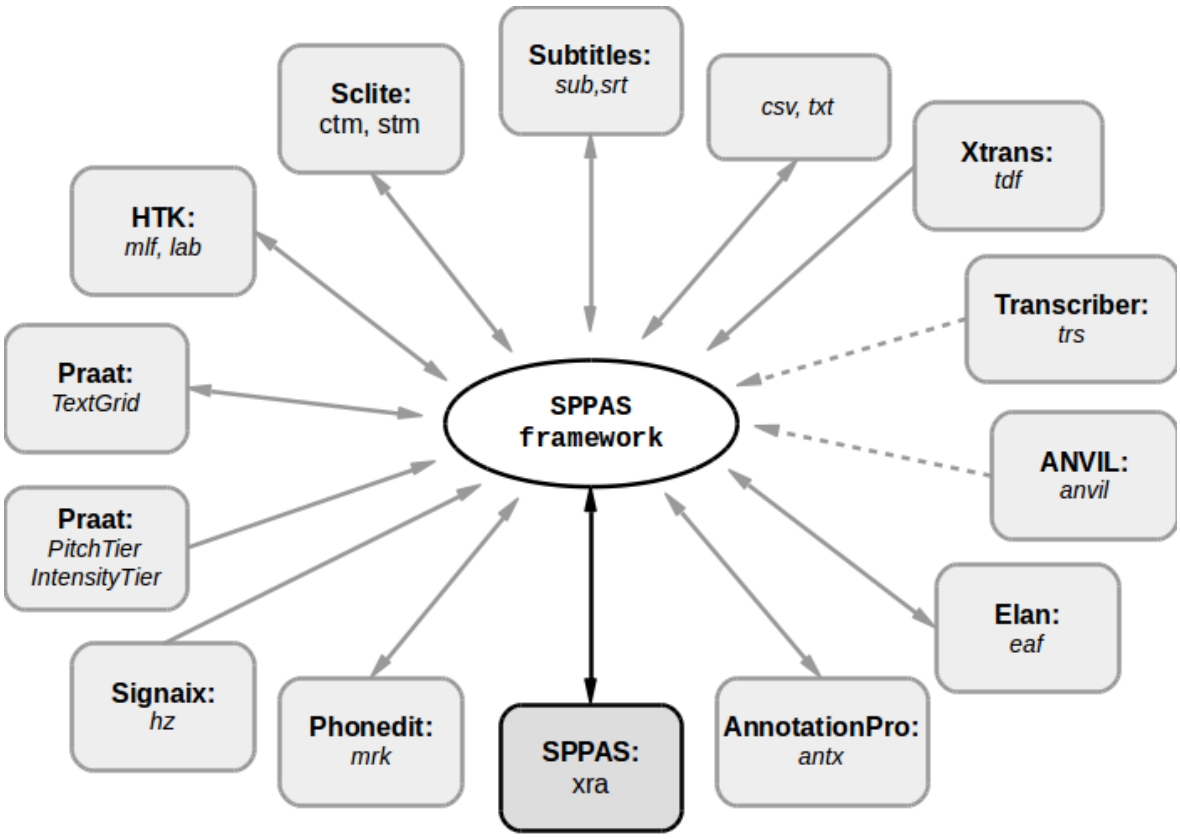


Figure 1.3: SPPAS conversion method and formats

- Annotation Pro: antx
- Phonedit: mrk
- Sclite: ctm, stm
- HTK: lab, mlf
- Subtitles: srt, sub
- Signaix: hz
- Spreadsheets, R,…: csv
- Audacity, notepads: txt

And the followings can be imported:

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

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.

1.5 About this documentation

This documentation 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.

2

Interface

2.1 Introduction

The current chapter describes how to use SPPAS with the Graphical User Interface (GUI) and with the Command-line User Interface (CLI). Each feature implemented in SPPAS corresponds to a program that can be invoked by the GUI or the CLI.

2.2 GUI Main frame

2.2.1 Launch SPPAS

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. In recent versions of MacOs X (e.g. from 10.11 El Capitan), 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 Finder/File Explorer, double-click on the `sppas.command` file.

The program will first check the version of wxpython and eventually ask to update. It will then check if the julius program can be accessed.

The main windows will open automatically. It is made of a menu (left), a title (top), the tips (middle), the list of files (left-middle), and the action buttons (right).

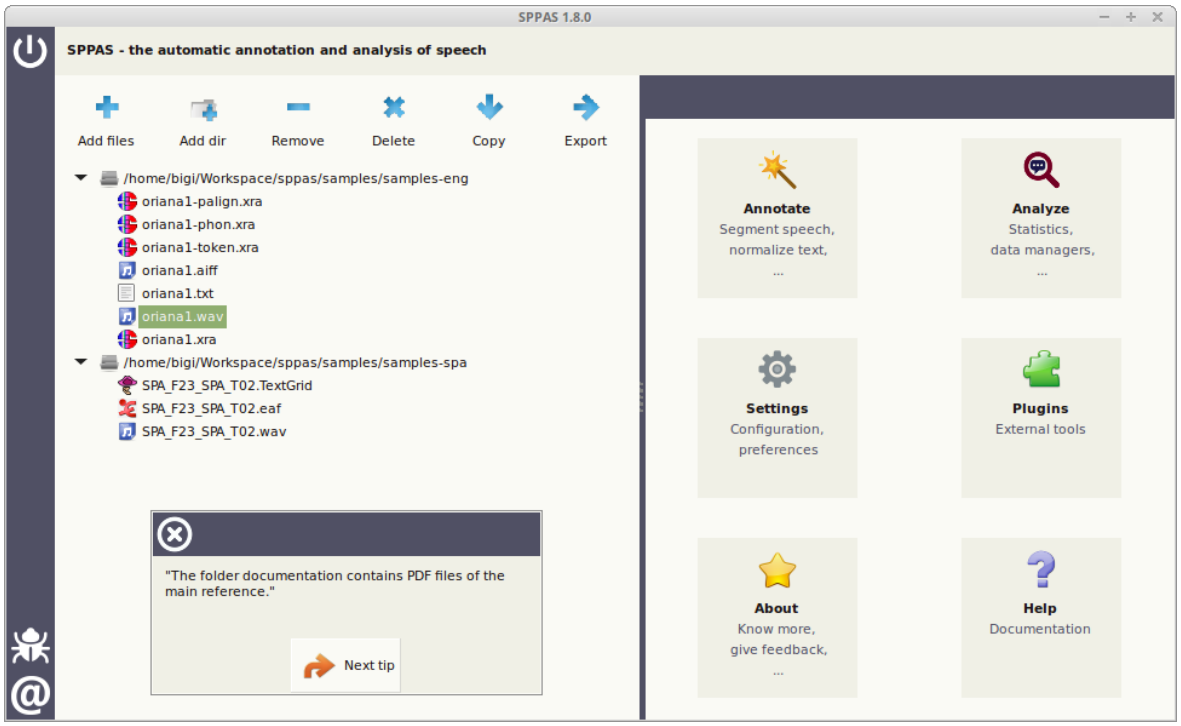


Figure 2.1: Main Frame of version 1.9.0

2.2.2 The tips

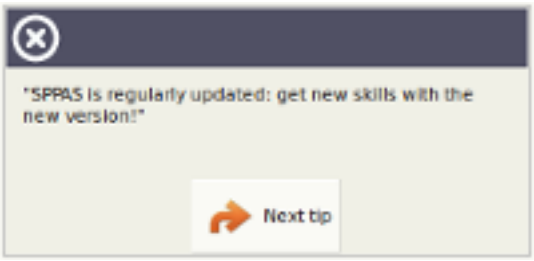


Figure 2.2: The tips included in the main frame

The frame includes message tips that are picked up randomly and printed to help users. Click on the button `Next Tip` to read another message, or click on the top-left button to close the tips frame. The `Settings` allows to show/hide tips at start-up. Anyone can suggest new tips to help other users. They have to be sent to the author by e-mail so that they will be included in the next version.

2.2.3 The menu

The menu is located at the left-side of the window. At top, a button allows to exit the program, and at bottom you can declare an issue or contact the author.

The `Exit` button closes all SPPAS frames properly.

To declare an issue, click on the bug button of the menu, then your default web browser will be opened at the appropriate URL. Take a quick look at the list of already declared issues and if any, click on the green button “New Issue”.

To contact the author, replace the text “Describe what you did here…” by your own comment, question or suggestion and choose how to send the e-mail: with your own default mailer, with gmail opened in your web browser, or by another e-mail client. Then, close the frame by clicking on the “Close” button.

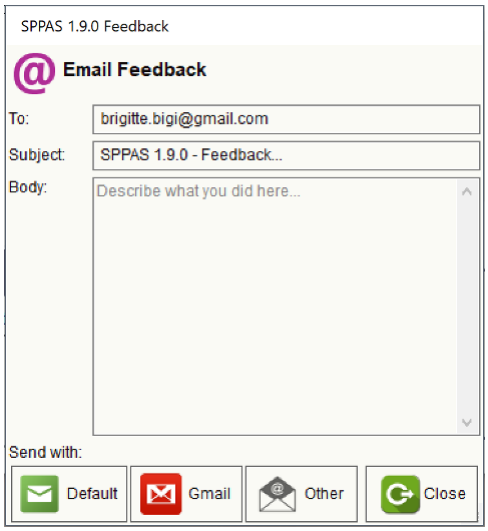


Figure 2.3: The Feedback Frame

2.2.4 The file explorer

The list contains directories and files the user added. However, only files that SPPAS can deal with, e.g. depending on the file extension, can be appended to the list. Files with an unknown extension are rejected.

> Remark: The files are added in the list, but they are not opened.

Select file(s)

To select a single file, all files of a directory or several files:

- a single click on a file name highlights and selects the chosen file. If other files or directories were previously selected, they are deselected.
- a single click on a directory name highlights the chosen directory and selects all files. If other files or directories were previously selected, they are deselected.
- to select several files or directories, the `ctrl` key (Linux/Windows) or `CMD` key (Apple) must be pressed while clicking each file or directory.
- to select several files or directories, the `shift` key can be pressed while clicking two files or directories.

Add file(s)

A single-click on the `Add files` button opens a window that allows to select the files to get. By default, only files with “.wav” extensions are proposed. When audio file(s) is/are selected, all files with the same name are automatically added into the file explorer.

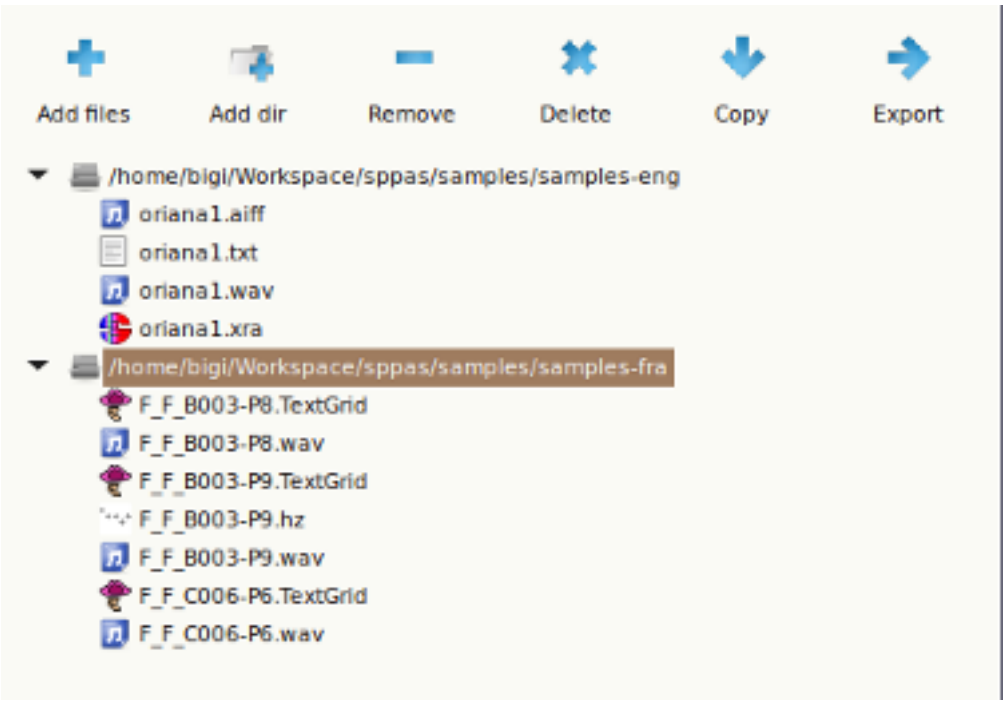


Figure 2.4: File explorer

The wildcard of this window can be changed to some other specific file extensions and then other files can be selected.

In both cases, only files with an appropriate extension will be added in the file explorer.

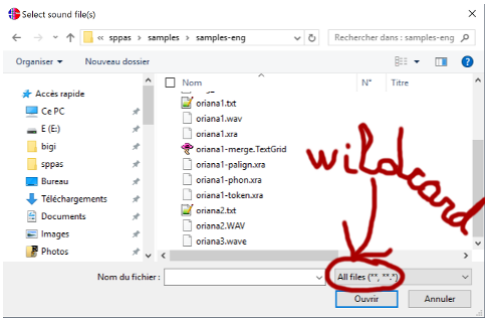


Figure 2.5: Adding specific files

Add a directory

A single-click on the `Add dir` button opens a window that allows to select the directory to get. Each audio file, and all related files - i.e. with the same name and with an appropriate extension, will be added into the file explorer.

Remove file(s)

A single-click on the `Remove` button removes the selected files/directories of the list. Notice that files are not deleted from the disk.

Delete file(s)

A single-click on the `Delete` button deletes *definitively* the selected files/directories of your computer, and remove them of the list. Notice that there is no way to get them back!

A dialogue window will open, and you have to confirm or cancel the definitive file(s) deletion.

Copy file(s)

A single-click on the `Copy` button allows to copy the selected file(s), change their name and/or change the location. It is also possible to change the file format by assigning the appropriate file extension.

Export file(s)

Export annotated files in an other format (csv, txt, …):

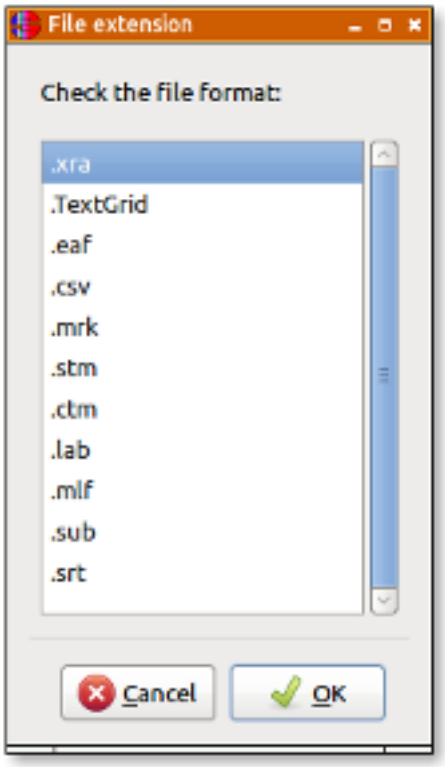


Figure 2.6: Export: Check the expected format in the list

After the export, a new window opens to report whether the file(s) were exported successfully or not. Actually, an export fails if: - the given file is not in UTF-8 encoding; - the given file contains several tiers and the output format supports only one; - the given file contains corrupted annotations.

2.2.5 Settings

To change user preferences, click on the `Settings` icon, then choose your preferred options in the tabs:

- General: fix background color, font color and font of the GUI, and enable/disable tips at start-up.

- Theme: fix how the icons of the GUI will look like.
- Annotations: fix automatic annotations parameters.

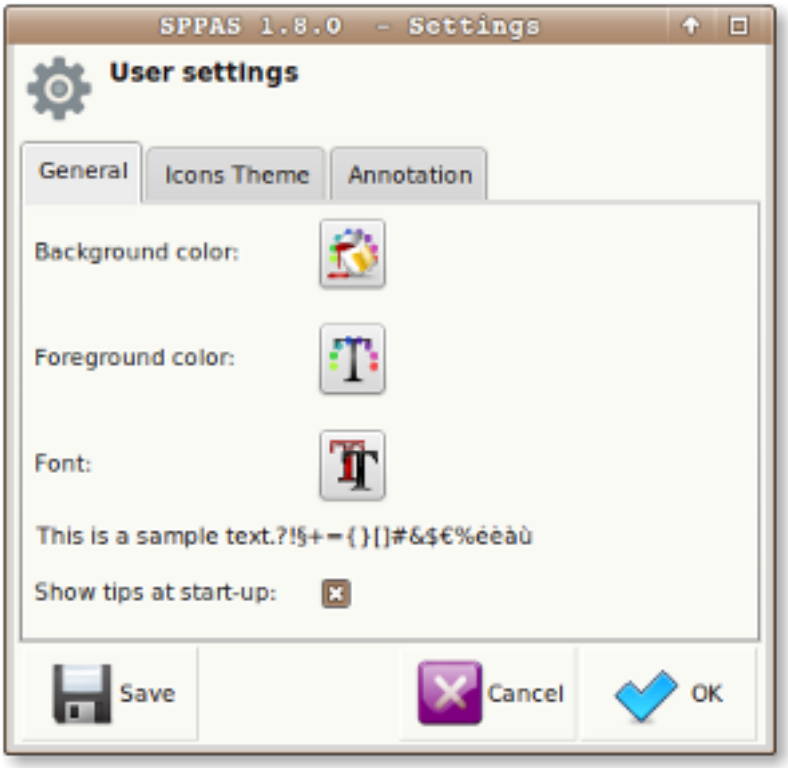


Figure 2.7: Settings is used to manage user preferences

These preferences can be saved in a file to be used each time SPPAS is executed. Finally, to close the settings window, click on:

- "Cancel" button to ignore changes;
- "Close" to apply changes then close the window.

2.2.6 About

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

2.2.7 Help

The `Help` action button opens the documentation and allows to browse in the chapters and sections.

2.2.8 Plugins

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.

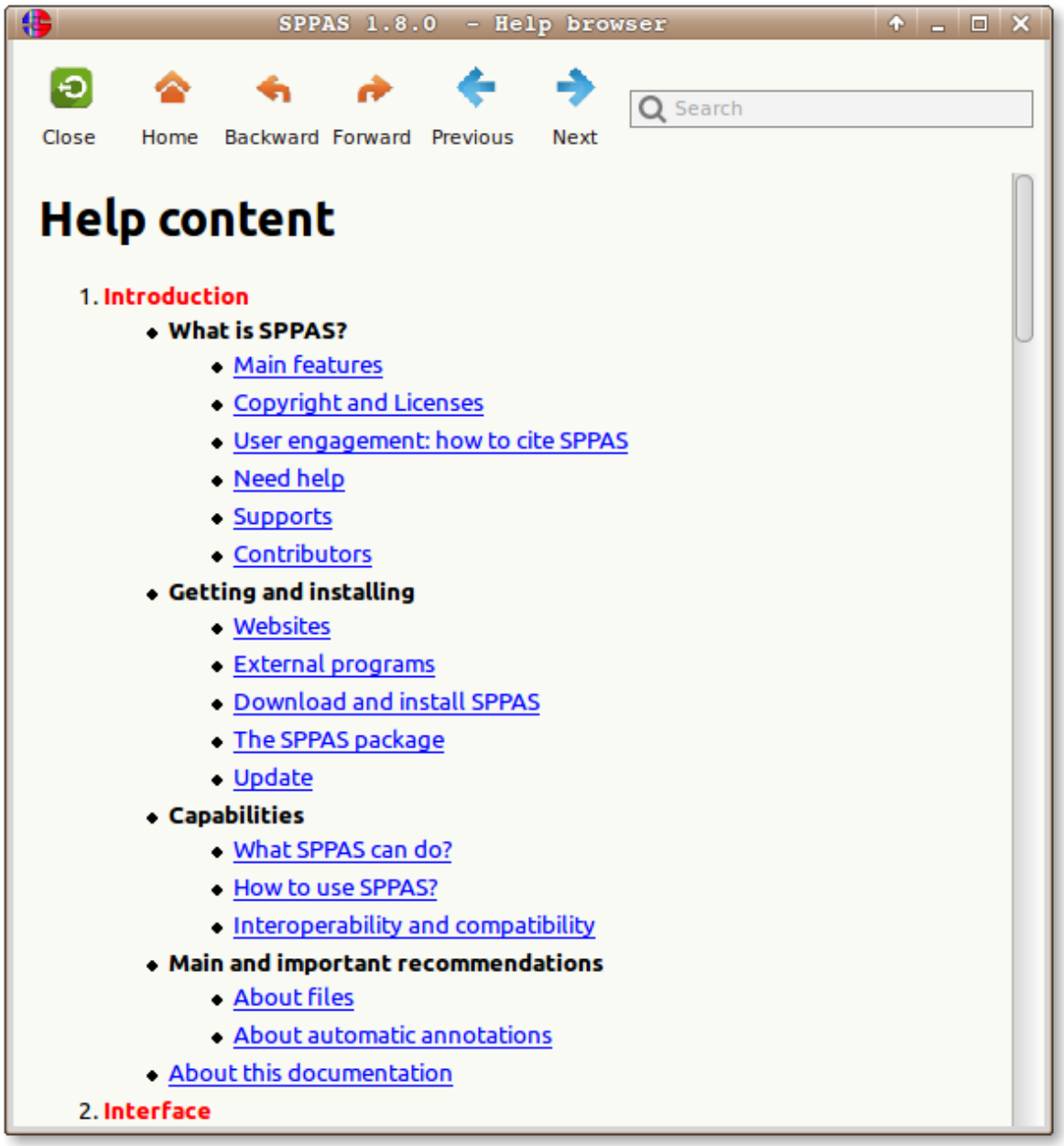


Figure 2.8: The Help browser window

The plugin system of SPPAS was fully changed at version 1.8.2; and plugins were all updated at version 1.9.0. Old plugins and new ones are not compatibles.

Installing a plugin

To install a plugin, simply follow this workflow:

1. 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.

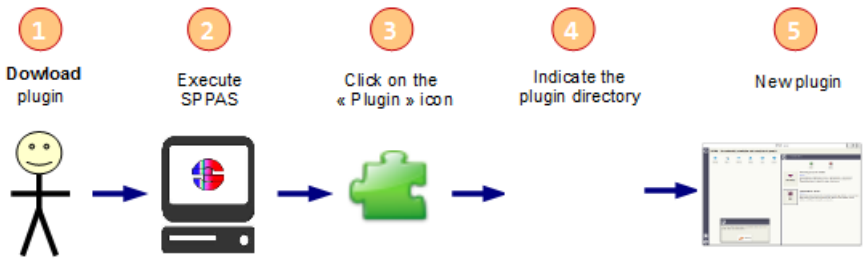


Figure 2.9: Installing a new Plugin

Deleting a plugin

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.

Using a plugin

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.

2.3 CLI

A command-line interface (CLI), also known as command-line user interface, 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 included in the SPPAS package. All these programs are written with the programming language Python 2.7.

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> annotation.py --help
```

There are 2 types of programs in the `sppas/bin` folder:

1. programs to execute an automatic annotation. These programs only requires Python 2.7 to be installed, and `julius` or `hvite` for the alignment program. They do not use a GUI so that installing wxPython is not required.
2. programs to execute an analysis tool. They take as argument the files to work on and they open the corresponding window. The GUI is launch.

In addition, the program `plugin.py` allows to install, execute and remove a plugin of SPPAS.

3

Automatic Annotations

3.1 Introduction

3.1.1 About this chapter

This chapter is not a description on how each automatic annotation is implemented and how it's working: references are available for that in chapter 7.

Instead, this chapter describes how each automatic annotation can be used in SPPAS, i.e. what is the goal of the annotation, what are the requirements, what kind of resources are used, what is the expected result and how to perform it within SPPAS.

Each automatic annotation process is illustrated as a workflow schema, where:

- blue boxes represent the name of the automatic annotation;
- red boxes represent tiers, with their name in white;
- green boxes indicate the resource;
- yellow boxes indicate the annotated file either given as input or produced as result.

3.1.2 Annotations methodology

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. 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, the first annotation to perform is IPUs segmentation. Indeed, at a first stage, the audio signal must be automatically segmented into Inter-Pausal Units (IPUs) which are blocks of speech bounded 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**. Then text normalization automatic annotation will normalize the orthographic transcription. The phonetization process will convert the normalized text in a set of pronunciations using X-SAMPA standard. Alignment will perform segmentation at phonemes and tokens levels, etc.

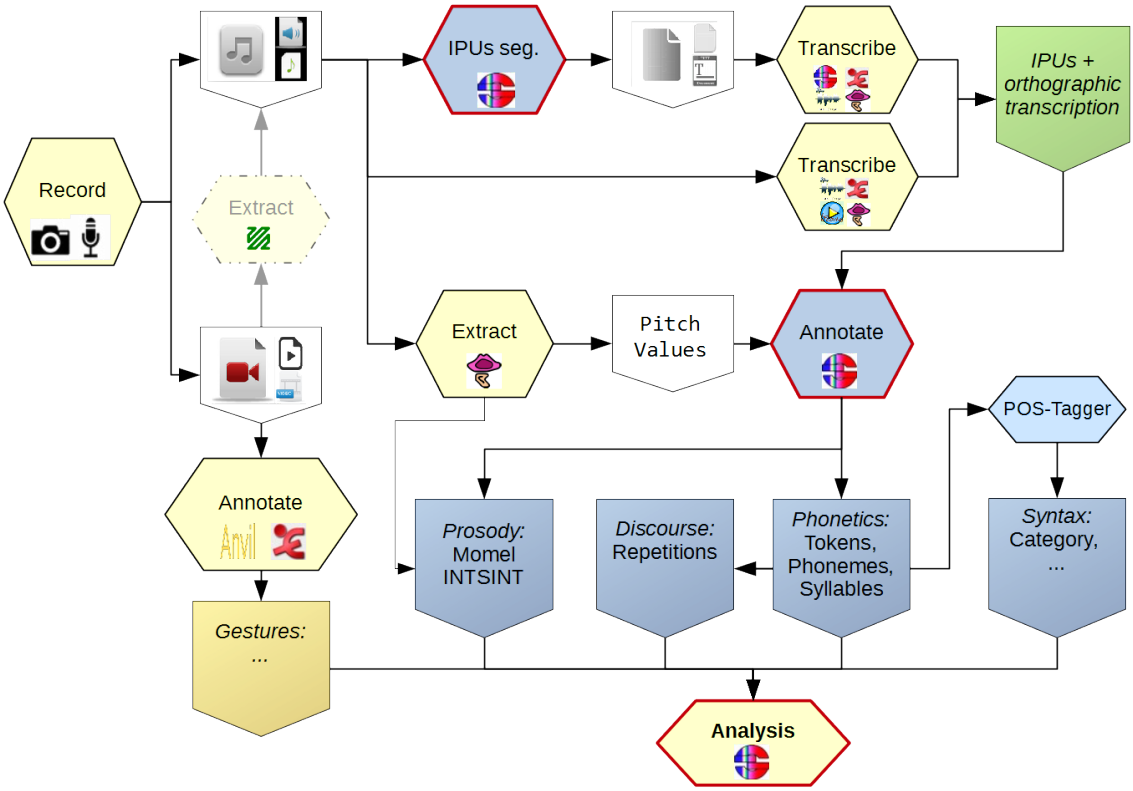


Figure 3.1: Annotation methodology

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

Among others, SPPAS is able to produce automatically annotations from a recorded speech sound and its orthographic transcription. Let us first introduce what is required in terms of files, and what is the exactly meaning of "recorded speech" and "orthographic transcription".

3.1.3 File formats and tier names

When using the Graphical User Interface, the file format for input and output can be fixed in the Settings and is applied to all annotations. However, while using the GUI to annotate, the file names of each annotation is already fixed and can't be changed.

When using the Command-Line interface, or when using scripts, each annotation can be configured independently (both file format and file names).

In all cases, **the name of the tiers are fixed and can't be changed**!

3.1.4 Recorded speech

Only `wav` and `au` audio file formats are supported by SPPAS.

Only mono audio files are supported by SPPAS.

SPPAS verifies if the audio file is 16 bits sample rate and 16000 Hz frame rate. Otherwise it automatically create a copy of the audio file converted to this configuration. For very long files, this process may take time. So, the following 2 options are possible:

1. be patient;
2. prepare by your own the required wav/mono/16000Hz/16bits files to be used in SPPAS.

Secondly, a relatively good recording quality is expected. Providing a guideline or recommendation for that is impossible, because it depends on many factors. For example, "IPU segmentation" requires 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. SPPAS simply 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…

3.1.5 Automatic Annotations with GUI

To perform automatic annotations with SPPAS Graphical User Interface, select first the list of audio files and/or a directory, then click on the "Annotate" button.

1. Enable each annotation to perform by clicking on the button in red. It will be turned green.

2. Select options and configure each annotation by clicking on the "Configure…" link text in blue.

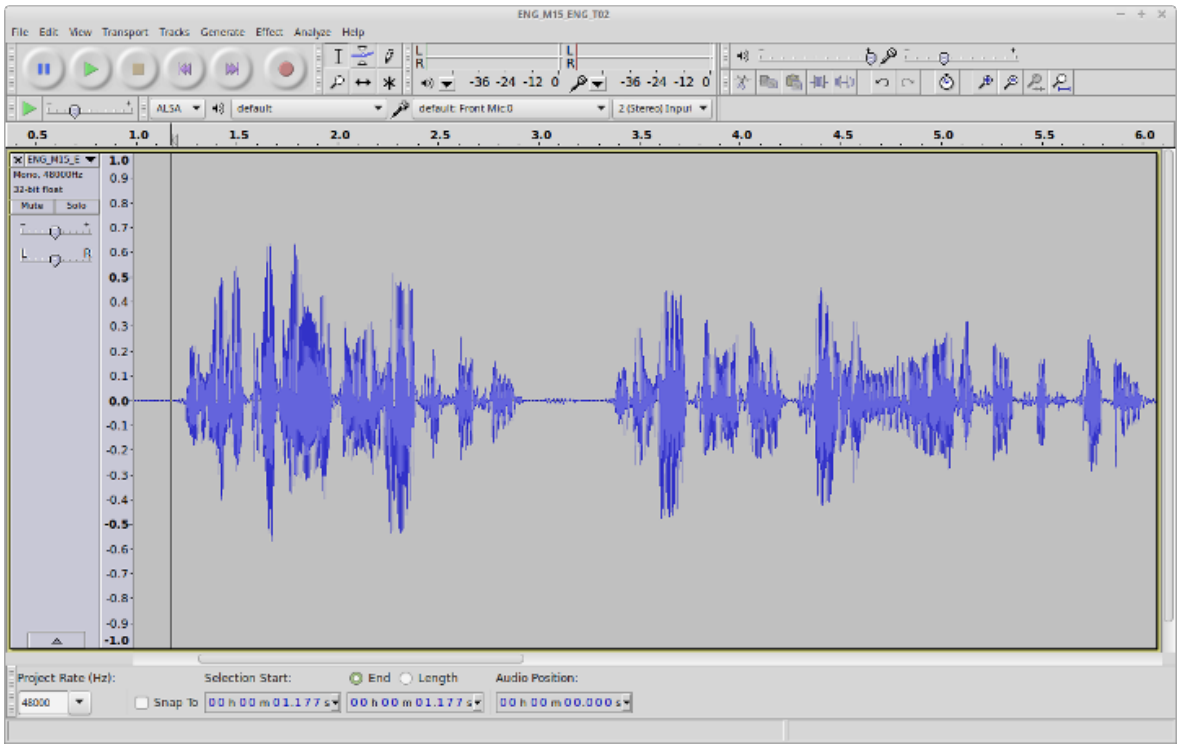


Figure 3.2: Example of expected recorded speech

3. Select the language for all annotations in one; or for each annotation independently by clicking on the "chains" button first then selecting each language.

4. Click on the *Perform annotations* button, and wait. Be patient! Particularly for Text Normalization or Phonetization: loading resources (lexicons or dictionaries) can be very long. Sometimes, the progress bar does not really show the progress… it depends on the operating system and the power of the computer. So, just wait!

5. It is important to read the Procedure Outcome report to check that everything happened normally during the automatic annotations.

3.1.6 Automatic Annotations with CLI

To perform automatic annotations with SPPAS Command-line User Interface, there is a main program `annotation.py`. Each annotation has also its own program with more options than the previous one.

This main 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. All annotations are pre-configured: no specific option can be specified. The options of annotations are fixed files with extension `.ini` in `sppas/etc` folder.

```
usage: annotation.py -w file|folder [options]


optional arguments:
   -h, --help       show this help message and exit
   -w file|folder   Input wav file name, or folder
```

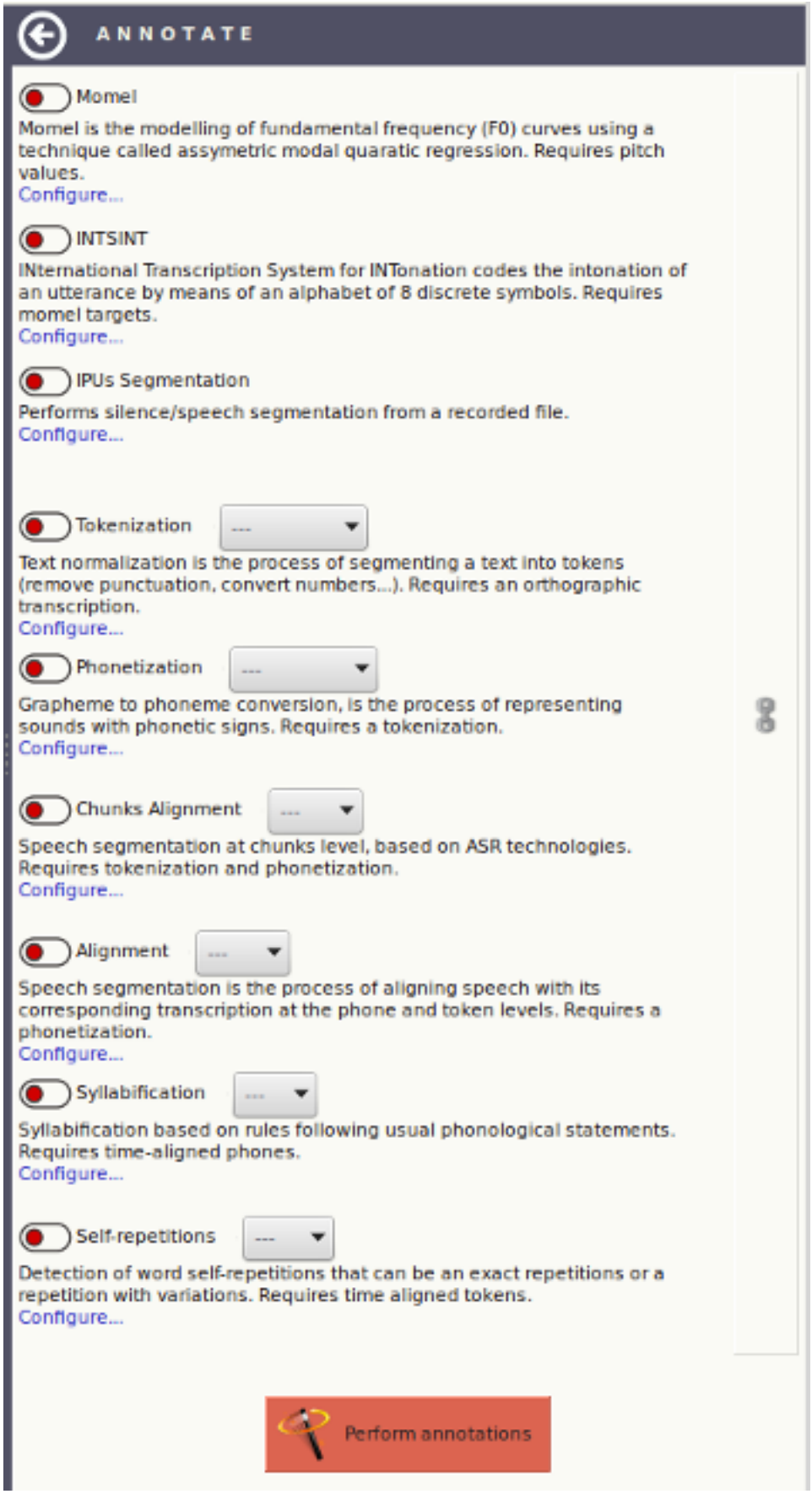


Figure 3.3: The annotate panel

```
-l lang          Input language, using iso639-3 code
-e extension     Output extension. One of: xra, textgrid, eaf, csv, ...
--momel          Activate Momel and INTSINT
--ipu            Activate IPUs Segmentation
--tok            Activate Text Normalization
--phon           Activate Phonetization
--align          Activate Alignment
--syll           Activate Syllabification
--rep            Activate Repetitions
--all            Activate ALL automatic annotations
--merge          Create a merged TextGrid file, if more than two automatic
                 annotations (this is the default).
--nomerge        Do not create a merged TextGrid file.
```

Examples of use:

```
./sppas/bin/annotation.py -w ./samples/samples-eng
                          -l eng
                          --ipu --tok --phon --align
```

A progress bar is displayed for each annotation. At the end of the process, a message indicates the name of the procedure outcome report file, which is `./samples/samples-eng.log` in our example. This file can be opened with any text editor (as Notepad++, vim, TextEdit, …).

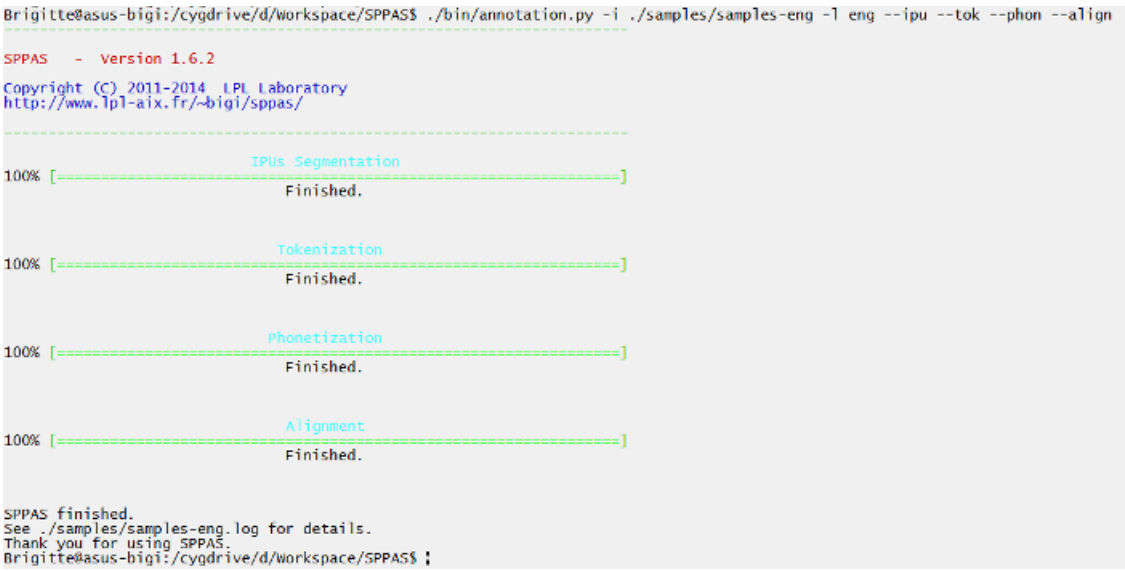


Figure 3.4: CLI: annotation.py output example

3.1.7 The procedure outcome report

It is very important to read conscientiously this report: it mentions exactly what happened during the automatic annotation process. This text can be saved: it is recommended to be kept it with the related data because it contains information that are interesting to know for anyone using the annotations.

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 1.9.0
Copyright (C) 2011-2017 Brigitte Bigi
Site web: http://www.sppas.org/
Contact: Brigitte Bigi(brigite.bigi@gmail.com)
```

Secondly, the text mentions 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 list of files to be annotated.
3. the list of annotations and if each annotation was activated or disabled. In that case, activated means that the checkbox of the AAP was checked by the user and that the resources are available for the given language. On the contrary, disabled means that either the checkbox of the AAP was not checked or the required resources are not available.
4. the file format of the resulting files.

Example:

```
Date: 2017-07-05 12:15:40.637000
Langages selectionnés:
  - Momel:
  - INTSINT:
  - IPUs Segmentation:
  - Text Normalization: eng
  - Phonetization: eng
  - Alignment: eng
  - Syllabification: und
  - Self-repetitions: eng
--------------------------------------------------------------------------------
Fichiers sélectionnés:
  - oriana1.wav
--------------------------------------------------------------------------------
Annotations sélectionnées:
  - Momel: désactivé
  - INTSINT: désactivé
  - IPUs Segmentation: sélectionné
  - Text Normalization: sélectionné
  - Phonetization: sélectionné
  - Alignment: sélectionné
  - Syllabification: désactivé
  - Self-repetitions: désactivé
--------------------------------------------------------------------------------
Extension de fichiers: .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 du fichier oriana1.wav
...  ... Options:
...  ...  ...  - faked: True
...  ...  ...  - std: False
...  ...  ...  - custom: False
```

Then, a diagnosis of the given file is printed. This latter can be: 1. "Valid": the file is relevant 2. "Admit": the file is not as expected but SPPAS will convert it in a copy and work on it. 3. "Invalid": SPPAS can't work with that file. The annotation of this file is disabled. In case 2 and 3, a message indicates the origin of the problem.

Example of "Valid" diagnosis:

```
...  ... Diagnostic:
...  ...  ...  - oriana1.xra: Valide.
```

Example of "Admit" diagnosis:

```
...  ... Diagnostic:
...  ...  ...  - M_3.wav: Admis. La taux d'échantillonage d'un fichier audio est
de préférence 16000 Hz. Ce fichier est échantilloné à 44100 Hz, SPPAS créera une
 copie et travaillera sur celle-ci.
...  ...  ...  - M_3-phon.xra: Valide.
...  ...  ...  - M_3-token.xra: Valide.
```

Then, if any, the annotation procedure prints message. 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 still performed successfully.
4. "[ERROR]" means that something happened abnormally and SPPAS failed to found a solution. The annotation was either not performed, or performed with a bad 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- est absent du dictionnaire et a été phonétisé
 automatiquement S-o-R-t
```

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

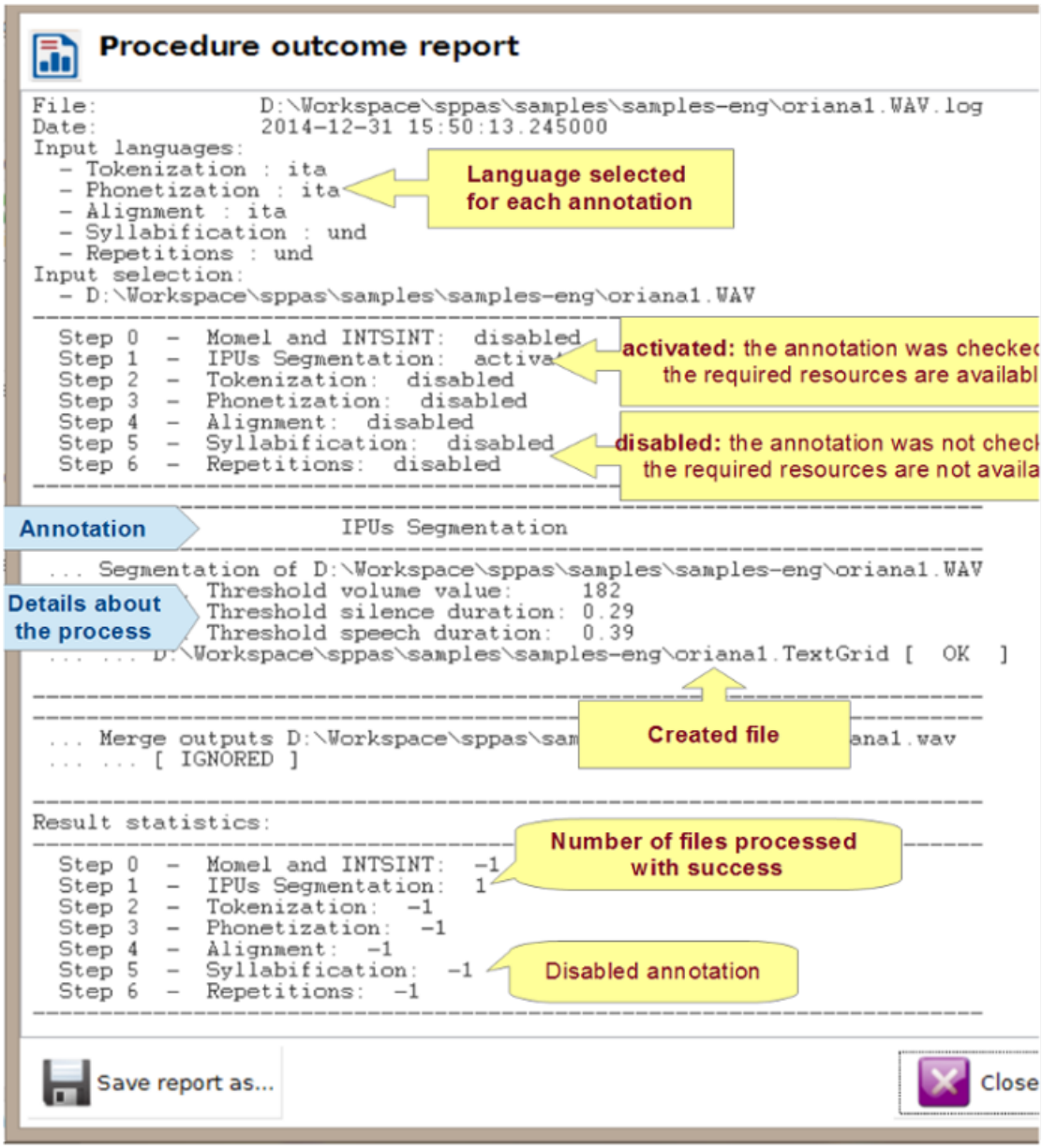


Figure 3.5: Procedure outcome report

3.2 New language support

Some of the annotations are requiring linguistic resources in order to work efficiently on a given language: text normalization requires a lexicon, phonetization requires a pronunciation dictionary, etc. Each section of this chapter which is describing an annotation is also including *the way to create the related resource*. The next chapter contains details about the existing resources - list of phonemes, authors, licenses, etc.

While starting SPPAS, the Graphical User Interface dynamically creates the list of available languages of each annotation by exploring the related folder. This means that:

- appended resources 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 appended while it was already being running.

3.3 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. *Transcription conventions* are then designed to provide rules for writing speech corpora. These conventions establish which are the phenomena to transcribe and also how to mention them in the orthography.

In speech, many phonetic variations occur. Some of these phonologically known variants are predictable and can be included in the pronunciation dictionary but many others are still unpredictable like invented words, regional words or words borrowed from another language. These specifics have a direct consequence on the automatic phonetization procedure (Bigi 2012). As a consequence, from the beginning of its development it was considered to be essential for SPPAS to deal with an **Enriched Orthographic Transcription**.

The transcription convention is summarized below. Notice that all the symbols must be surrounded by whitespace. The file `TOE-SPPAS.pdf` available in the `documentation` folder gives details of the convention, including examples of what is recommended.

Convention overview:

- truncated words, noted as a '-' at the end of the token string (an ex- example);
- noises, noted by a '*' (not available for Japanese and Cantonese);
- laughs, noted by a '@' (not available for Japanese and Cantonese);
- short pauses, noted by a '+';
- elisions, mentioned in parenthesis;
- specific pronunciations, noted with brackets [example,eczap];
- comments are noted inside braces or brackets without using comma {this} or [this and this];
- liaisons, noted between '=' (an =n= example);
- morphological variants with <like,lie ok>,
- proper name annotation, like $ John S. Doe $.

SPPAS also allows to include in the transcription the regular punctuations, and numbers (not available for some languages). Numbers will be automatically converted to their written form during Text Normalization process.

From this Enriched Orthographic construction, several derived transcriptions can be generated automatically, including the followings:

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

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

- standard: *this is + hum an enriched transcription number one*
- faked: *this iz + hum an enrich transcription number one*

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

- filled pauses;
- short pauses;
- repeats;
- noises and laugh items (not available for: English, Japanese and Cantonese).

Finally, it has to be noticed that this convention is not software-dependent. The orthographic transcription can be performed with IPUscriber tool of SPPAS, Praat, Annotation Pro, Audacity, …

3.4 Inter-Pausal Units (IPUs) segmentation

3.4.1 Overview

After recording, the first annotation to perform is IPUs segmentation. At the early first stage, the Inter-Pausal Units (IPUs) must be automatically detected in the audio signal. IPUs are blocks of speech bounded by silent pauses of more than X ms. An annotated file with the bounds of IPUs is then created. This segmentation should be verified manually: the better recording quality, thus the better IPUs segmentation.

The "IPUs segmentation" automatic annotation can perform 3 actions:

1. find silences/ipus of a recorded file.
2. find silences/ipus of a recorded file when the utterances are given in a `txt` file.
3. split/save a recorded file into multiple files, depending on segments indicated in a time-aligned file.

Notice that all time values are indicated in seconds.

3.4.2 Silence/Speech segmentation

The IPUs Segmentation is a semi-automatic annotation process. It performs a silence detection from a recorded file. This segmentation provides an annotated file with one tier named "IPU". The silence intervals are labelled with the "#" symbol, and IPUs intervals are labelled with "ipu_" followed by the IPU number.

The following parameters must be properly fixed:

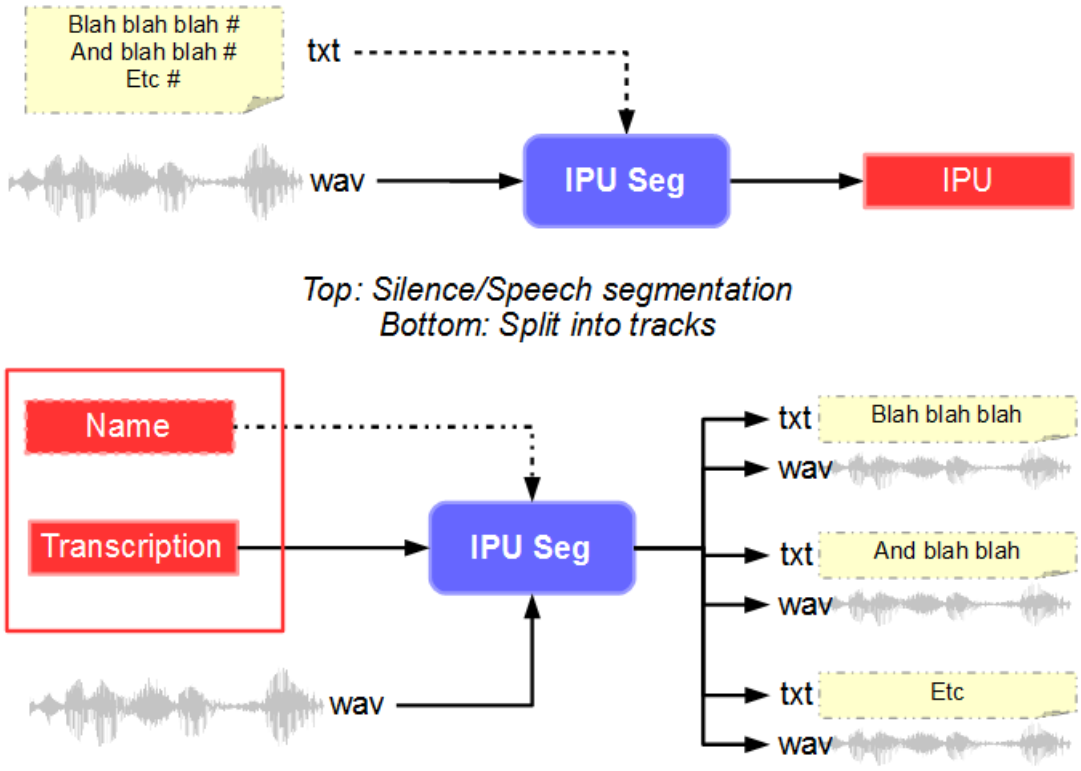


Figure 3.6: IPU Segmentation workflow

- 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 following options are also available to configure the annotation: * Add the IPU index in each interval of the Transcription tier * Split the audio file into tracks (only if silence/ipus segmentation is already available) * Save tracks as annotation files (only if silence/ipus segmentation is already available)

The procedure outcome report indicates the values (volume, minimum durations) that was used by the system for each sound file given as input. It also mentions the name of the result file.

```
  ... IPUs Segmentation of file oriana1.wav
... ... [  INFO  ] A transcription was found, perform Silence/Speech segmentation
  time-aligned with a transcription oriana1.txt
```

```
...   ... Options:
...   ...   ...   - save_as_trs: False
...   ...   ...   - dirtracks: False
...   ...   ...   - addipuidx: True
...   ... Diagnostic:
...   ...   ...   - oriana1.wav: Valide.
...   ...   ... Threshold volume value:     122
...   ...   ... Threshold silence duration: 0.3
...   ...   ... Threshold speech duration:  0.5
...   ... [     OK     ] oriana1.xra
```

The resulting annotated file has to be checked manually. It can be viewed with the SPPAS Visualizer tool, or it can be viewed/revised with Praat.

If the option values were not relevant enough: delete the annotated file that was previously created, change such values and re-annotate.

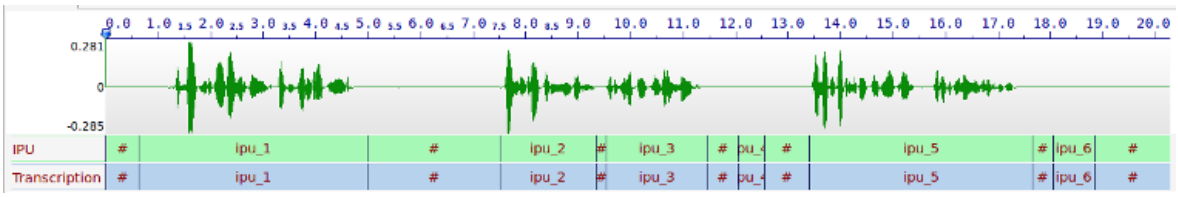


Figure 3.7: Result of IPUs Segmentation: silence detection

Notice that the speech segments can be transcribed using the SPPAS "IPUScriber" tool.

3.4.3 Silence/Speech segmentation time-aligned with a transcription

Inter-Pausal Units segmentation can also consist in aligning macro-units of a document with the corresponding sound.

SPPAS identifies silent pauses in the signal and attempts to align them with the utterances proposed in the transcription file, under the assumption that each such unit is separated by a silent pause. 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 segmentation provides an annotated file with one tier named "IPU". The silence intervals are labelled with the "#" symbol, as ipus intervals 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.

> Important: This segmentation was tested on documents no longer than one paragraph (about 1 minute speech).

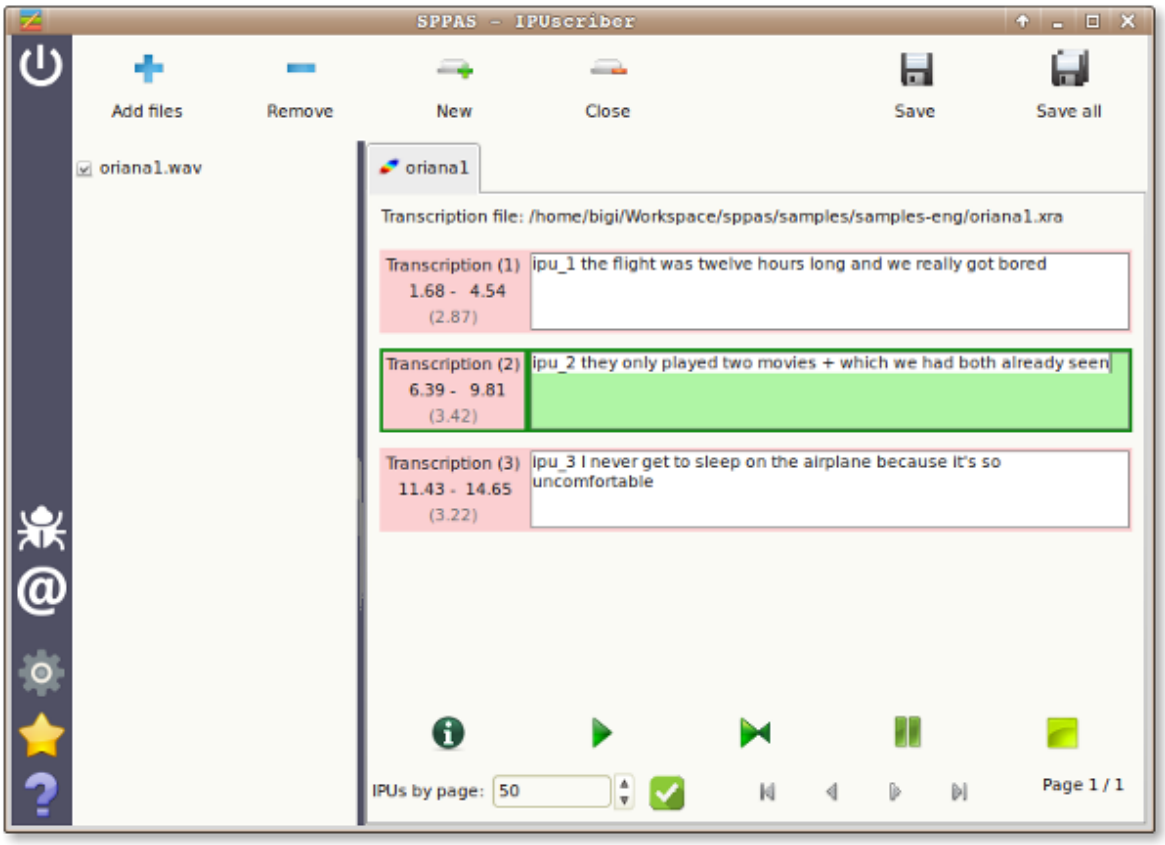


Figure 3.8: Orthographic transcription based on IPUs Segmentation

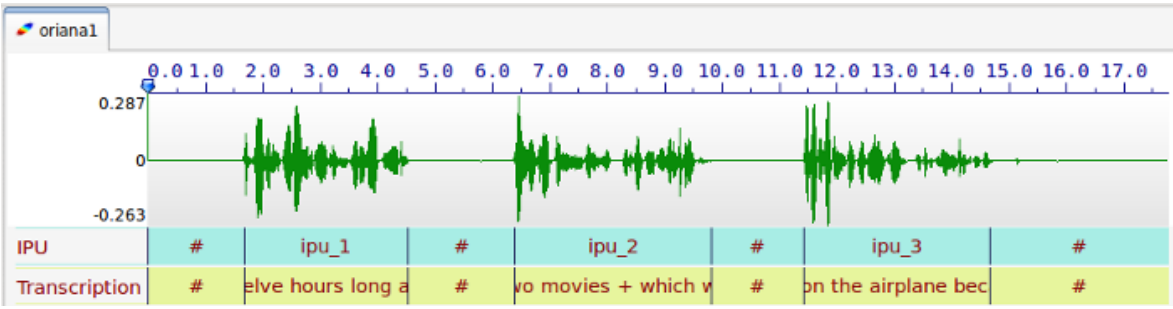


Figure 3.9: Silence/Speech segmentation with orthographic transcription

3.4.4 Split into multiple files

IPUs segmentation can split the sound into multiple files (one per IPU), and it creates a text file for each of the tracks. By default, the names of the output files are “track_0001”, “track_0002”, etc. Optionally, if the input annotated file contains a tier with name exactly “Name”, then the content of this tier will be used to fix output file names.

Figure 3.10: Data to split

In the example above, the automatic process will create 6 files: FLIGTH.wav, FLIGHT.txt, MOVIES.wav, MOVIES.txt, SLEEP.wav and SLEEP.txt. It is up to the user to perform another IPUs segmentation of these files to get another file format than txt (xra, TextGrid, …). Refer to the previous section “Silence/Speech segmentation time-aligned with a transcription”.

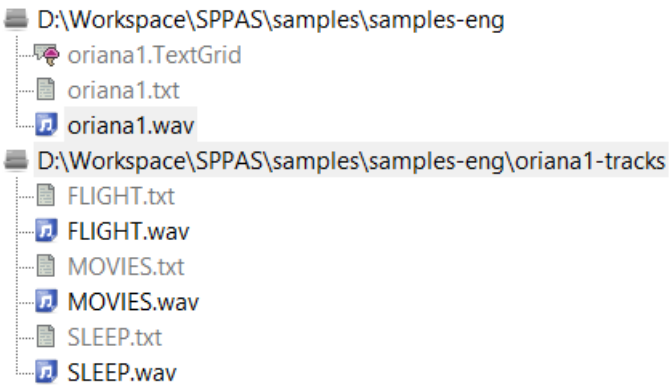


Figure 3.11: Data split

3.4.5 Perform IPUs Segmentation with the GUI

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

3.4.6 Perform IPUs Segmentation with the CLI

`wavsplit.py` is the program to perform the IPUs segmentation, i.e. silence/ipus segmentation.

```
usage: wavsplit.py -w file [options]
```

Generic options:

```
-w file         audio input file name
-d delta shift  Add this time value to each start
                bound of the IPUs
-D delta shift  Add this time value to each end
                bound of the IPUs
-h, --help      Show the help message and exit
```

Options that can be fixed for the silence/ipus segmentation:

```
-r float    Window size to estimate rms, in seconds
            (default value is: 0.010)
-m value    Drop speech shorter than m seconds long
            (default value is: 0.300)
-s value    Drop silences shorter than s seconds long
            (default value is: 0.200)
-v value    Assume that a rms lower than v is a silence
            (default value is: 0 which means to auto-adjust)
```

Options that can be fixed for the silence/ipus segmentation with a given orthographic transcription. It must be choose one of -t or -n options:

```
-t file     Input transcription file (default: None)
-n value    Input transcription tier number (default: None)
-N          Adjust the volume cap until it splits
            into nb tracks (default: 0=automatic)
```

Output options:

```
-o dir      Output directory name       (default: None)
-e ext      Output tracks extension     (default: txt)
-p file     File with the segmentation  (default: None)
```

Examples of use to get each IPU in a separate wav file and its corresponding transcription in a textgrid file:

Step1: find silences/ipus ~~~~ ./sppas/bin/wavsplit.py -w ./samples/samples-eng/oriana1.wav -d 0.01 -D 0.01 -t ./samples/samples-eng/oriana1.txt -p ./samples/samples-eng/oriana1.xra ~~~~

Step2: split ipus into individual tracks ~~~~ ./sppas/bin/wavsplit.py -w ./samples/samples-eng/oriana1.wav -t ./samples/samples-eng/oriana1.xra -o ./samples/samples-eng/oriana1 -e textgrid ~~~~

3.5 Text normalization

3.5.1 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.

3.5.2 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.

3.5.3 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.

3.5.4 Perform Text Normalization with the GUI

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.

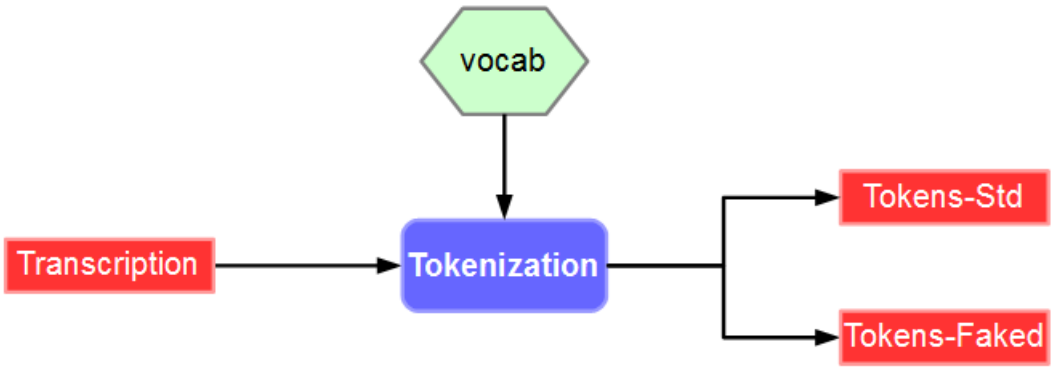


Figure 3.12: Text normalization workflow

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.

3.5.5 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 -r vocab [options]

optional arguments:
    -r vocab         Vocabulary file name
    -i file          Input file name
    -o file          Output file name
  --delimiter char Use a delimiter character instead of a space for word delimiter.
    -h, --help       Show the help message and exit
```

The following situations are possible:

1. no input is given: the input is `stdin` and the output is `stdout` (if an output file name is given, it is ignored). Only the normalization with a faked orthography is printed.

2. an input and an output are given: the output file is created (or erased if the file already exists) and the result of the process is added to this file..

Example of use, using stdin/stdout:

```
$ echo "The te(xt) to normalize 123." |\
  ./sppas/bin/normalize.py
  -r ./resources/vocab/eng.vocab
$ the te to normalize one_hundred_twenty-three
```

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

Example of use on a transcribed file:

```
$ ./sppas/bin/normalize.py -r ./resources/vocab/eng.vocab
  -i ./samples/samples-eng/oriana1.xra
  -o ./samples/samples-eng/oriana1-token.xra
```

3.6 Phonetization

3.6.1 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.

3.6.2 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`

3.6.3 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.

3.6.4 Perform Phonetization with the GUI

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 "Phonetization".

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

3.6.5 Perform Phonetization with the CLI

`phonetize.py` is the program to perform Phonetization on a given file.

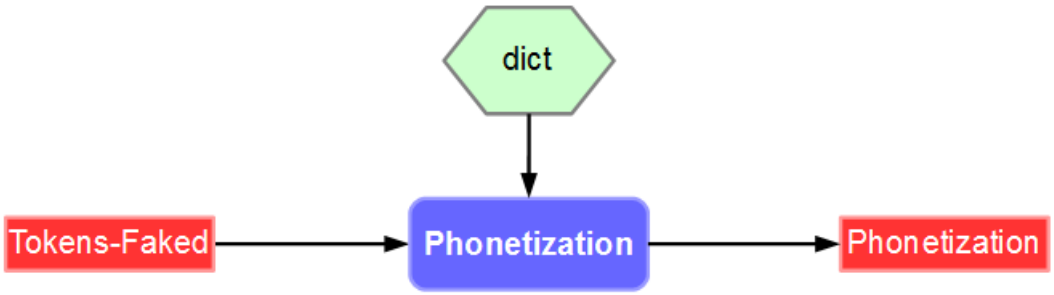


Figure 3.13: Phonetization workflow

```
usage: phonetize.py -r dict [options]

optional arguments:
    -r dict      Pronunciation dictionary file name (HTK-ASCII format)
    -m map       Pronunciation mapping table
    -i file      Input file name
    -o file      Output file name
    --nounk      Disable unknown word phonetization
    -h, --help   Show the help message and exit
```

Examples of use:

```
$ echo "the te totu" |\
 ./sppas/bin/phonetize.py
    -r ./resources/dict/eng.dict
    --nounk
$ D-@|D-V|D-i: t-i: UNK
$
$ echo "the te totu" |\
 ./sppas/bin/phonetize.py -r ./resources/dict/eng.dict
$ D-@|D-V|D-i: t-i: t-u-t-u|t-i-t-u|t-A-t-u|t-@-t-u
```

If we suppose that the previous text was read by a French native speaker, the previous example can be phonetized as:

```
$ echo "the te totu" |\
 ./sppas/bin/phonetize.py
     -r ./resources/dict/eng.dict
     -m ./resources/dict/eng-fra.map
$ D-@|z-@|v-@|z-9|D-V|v-9|v-V|D-9|z-V|D-i:|z-i|v-i|D-i|v-i:|z-i:
  t-i:|t-i
  t-u-t-u|t-i-t-u|t-A-t-u|t-@-t-u
```

Example of use on a tokenized file:

```
./sppas/bin/phonetize.py
-d ./resources/dict/eng.dict
```

```
-i ./samples/samples-eng/oriana1-token.xra
-o ./samples/samples-eng/oriana1-phon.xra
```

3.7 Alignment

3.7.1 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 the `HVite` command of HTK-Toolkit.

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 phoneme 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.

In addition, all acoustic models (except English, Japanese and Cantonese) include the following fillers: - dummy: un-transcribed speech - gb: garbage, for noises - @@: laughter

They allow SPPAS to time-align automatically these phenomena whose can be very frequent in speech, like laugh items (Bigi and Bertrand 2015). No other system is able to achieves this task!

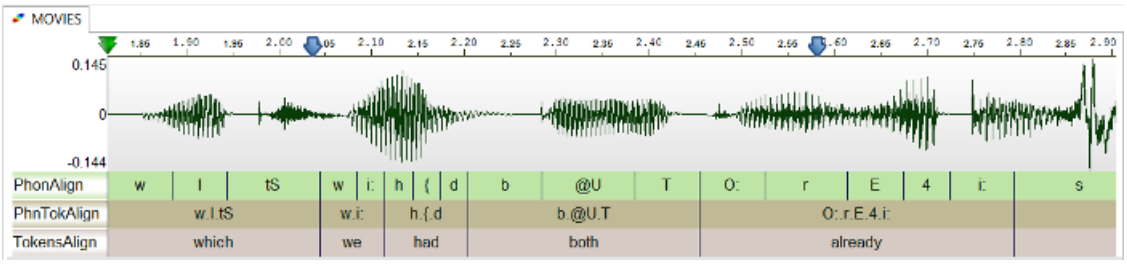


Figure 3.14: SPPAS alignment output example

3.7.2 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.

3.7.3 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.

3.7.4 Perform Alignment with the GUI

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).

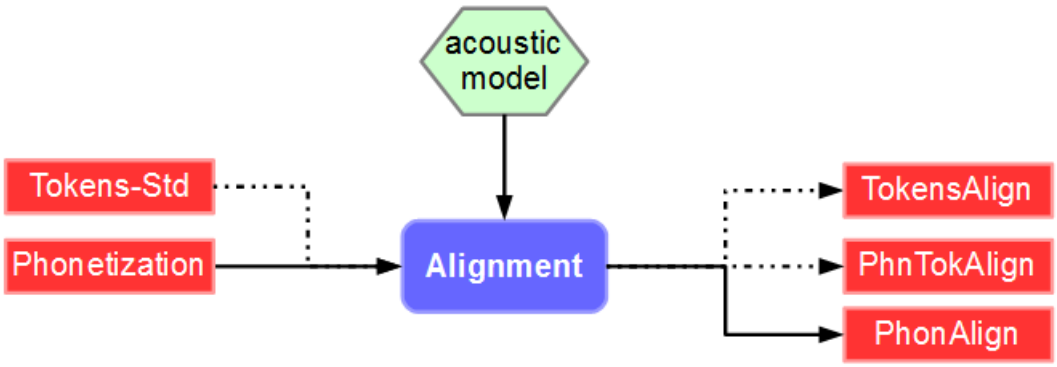


Figure 3.15: Alignment workflow

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 Activity tier will append another tier with activities as intervals, i.e. speech, silences, laughter, noises…

- create the PhnTokAlign will append anoter 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.

3.7.5 Perform Alignment with the CLI

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

```
usage: alignment.py -w file -i file -r file -o file [options]

optional arguments:
    -w file     Input wav file name
    -i file     Input file name with the phonetization
    -I file     Input file name with the tokenization
    -r file     Directory of the acoustic model of the language of the text
    -R file     Directory of the acoustic model of the mother language
                of the speaker
    -o file     Output file name with alignments
    -a name     Aligner name. One of: julius, hvite, basic (default: julius)
    --extend    Extend last phoneme/token to the wav duration
    --basic     Perform a basic alignment if error with the aligner
    --infersp   Add 'sp' at the end of each token and let the aligner
                to decide the relevance
    --noclean   Do not remove temporary data
    -h, --help  Show the help message and exit
```

Example of use:

```
./sppas/bin/alignment.py
-r ./resources/models/models-eng
-w ./samples/samples-eng/oriana1.WAV
-i ./samples/samples-eng/oriana1-phon.xra
-I ./samples/samples-eng/oriana1-token.xra
-o ./samples/samples-eng/oriana1-palign.xra
```

3.8 Syllabification

3.8.1 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 were established to deal with these classes.

Syllabification requires a configuration file to indicate phonemes, classes and rules.

Figure 3.16: Syllabification example

3.8.2 Adapt Syllabification

Any user can change the set of rules by editing and modifying the configuration file of a given language. 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 O`

The couples phoneme/class are made of 3 columns: the first column is the key-word PHONCLASS, the second column is the phoneme symbol, the third column is the class symbol. The constraints on this definition are:

- a pause is mentioned with the class-symbol #,
- a class-symbol is only one upper-case character, except:
  - the character X if forbidden;
  - 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 column 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).

3.8.3 Support of a new language

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

- "syllConfig-" followed by language name with iso639-3 standard
- extension ".syll"

3.8.4 Perform Syllabification with the GUI

The Syllabification 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 a tier containing the time-aligned phonemes 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 3 tiers: Syllables, Classes and Structures.

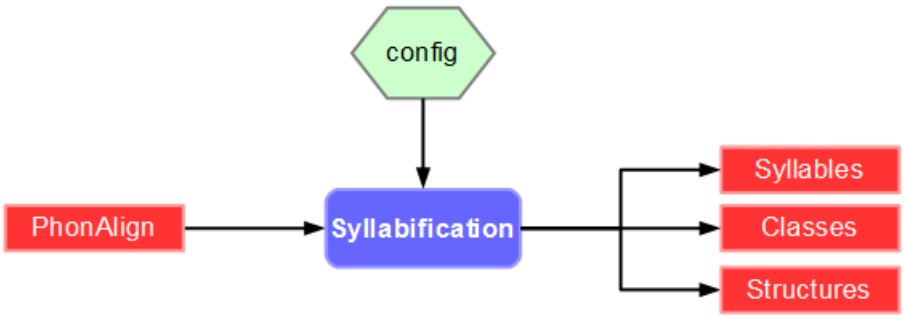


Figure 3.17: Syllabification workflow

Click on the Syllabification activation button, select the language and click on the "Configure…" blue text to fix options.

3.8.5 Perform Syllabification with the CLI

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

```
usage: syllabify.py -r config [options]
```

```
optional arguments:
    -r config   Rules configuration file name
    -i file     Input file name (time-aligned phonemes)
    -o file     Output file name
    -e file     Reference file name to syllabify between intervals
    -t string   Reference tier name to syllabify between intervals
  --nophn    Disable the output of the result that does not use the reference tier
    -h, --help   Show the help message and exit
```

3.9 Repetitions

3.9.1 Overview

This automatic detection focus on word repetitions, which can be an exact repetition (named strict echo) or a repetition with variation (named non-strict echo).

SPPAS implements *self-repetitions* and *other-repetitions* detection (Bigi et al. 2014). The system is based only on lexical criteria. The proposed algorithm is focusing on the detection of the source.

This process requires a list of stop-words, and a dictionary with lemmas but the system can process without them.

3.9.2 Adapt Repetitions and support of a new language

The result is significantly better if the right list of stop-words and a good dictionary of lemmas are available. Both are located in the "vocab" folder of the "resources" directory respectively with ".stp" and ".lem" extensions. These files are with UTF-8 encoding and "LF" for newline.

3.9.3 Perform Repetitions with the GUI

The Graphical User Interface only allows to detect self-repetitions.

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 for other-repetitions. The annotation provides one annotated file with 2 tiers: Sources and Repetitions.

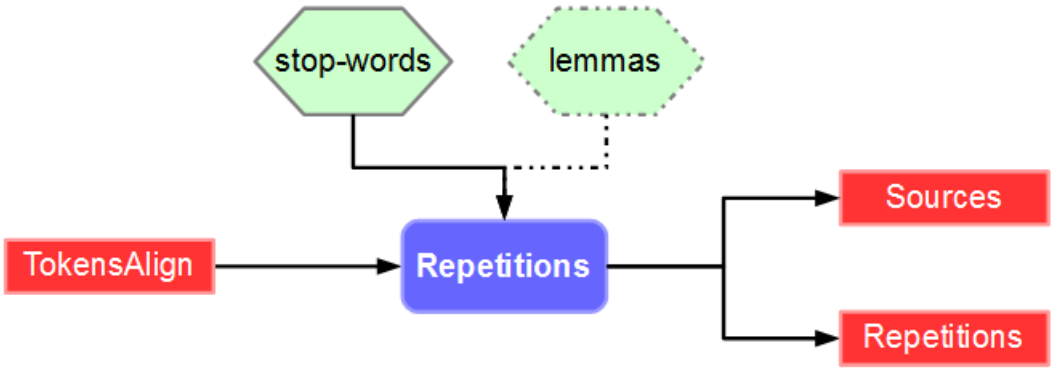


Figure 3.18: Repetition detection workflow

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

3.9.4 Perform Repetitions with the CLI

`repetition.py` is the program to perform automatic detection of self-repetitions or other-repetitions if data of a second speaker is given.

```
usage: repetition.py -i file [options]

optional arguments:
  -h, --help  show this help message and exit
  -i file     Input file name with time-aligned tokens of the self-speaker
  -r folder   Directory with resources
  -l lang     Language code in iso639-3
  -I file     Input file name with time-aligned tokens of the echoing-speaker
              (if ORs)
  -o file     Output file name
```

3.10 Momel and INTSINT

3.10.1 Momel (modelling melody)

Momel is an algorithm for the automatic modeling of fundamental frequency (F0) curves using a technique called asymetric 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 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’

3.10.2 Encoding of F0 target points using the “INTSINT” system

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 (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.

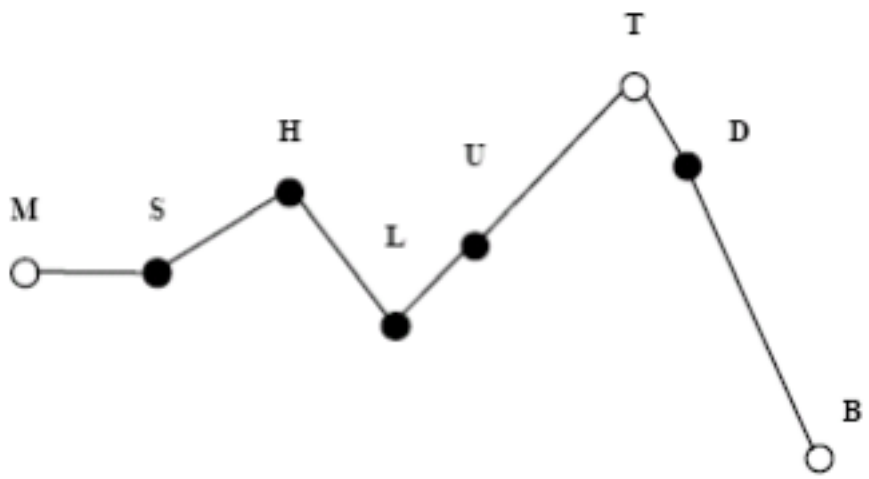
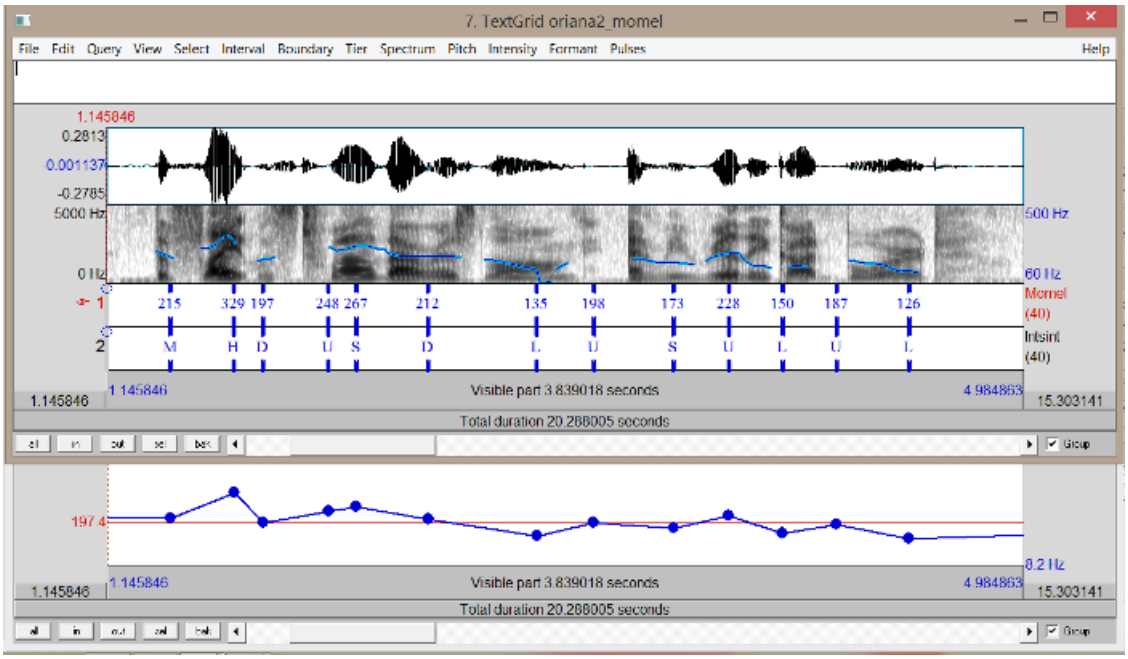


Figure 3.19: INTSINT example

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.



3.10.3 Perform Momel and INTSINT with the GUI

Click on the Momel activation button, select the language and click on the "Configure…" blue text to fix options. And click on the INTSINT activation button.

3.10.4 Perform Momel and INTSINT with the CLI

`momel-intsint.py` is the program perform both momel and INTSINT.

```
usage: momel-intsint.py [options] -i file

optional arguments:
  -i file              Input file name (extension: .hz or .PitchTier)
  -o file              Output file name (default: stdout)
  --win1 WIN1          Target window length (default: 30)
  --lo LO              f0 threshold (default: 50)
  --hi HI              f0 ceiling (default: 600)
  --maxerr MAXERR      Maximum error (default: 1.04)
  --win2 WIN2          Reduct window length (default: 20)
  --mind MIND          Minimal distance (default: 5)
  --minr MINR          Minimal frequency ratio (default: 0.05)
  --non-elim-glitch
  -h, --help           Show the help message and exit
```

4

Resources for Automatic Annotations

4.1 Overview

Automatic annotations included in SPPAS are implemented with language-independent algorithms. This means that adding a new language into SPPAS only consists in adding resources related to the annotation (like lexicons, dictionaries, models, set of rules, etc).

All resources can be edited, modified, changed or deleted by users.

To know what the resources are for and other information, refer to chapter 3 of this documentation: Each resource is used by an automatic annotation.

The resources are language dependent and the name of the files are based on the ISO639-3 international standard. See http://www-01.sil.org/iso639-3/ for the list of all languages and codes.

In the next sections, the table indicates the list of phonemes that are included in the resources required for phonetization, alignment and syllabification of a given language. The first column represents the symbols used by SPPAS and the other columns are intended to help users to better understand what it means.

The encoding of phonemes in SPPAS resources is based on a computer-readable phonetic list of 7-bit printable ASCII characters: X-SAMPA. This Extended Speech Assessment Methods Phonetic Alphabet was developed in 1995 by John C. Wells, professor of phonetics at the University of London. X-SAMPA is a language-independent notation covering the entire International Phonetic Alphabet - IPA repertoire. A plugin allows to convert time-aligned phonemes from X-SAMPA to IPA.

The acoustic models created by the author, Brigitte Bigi, were trained using the HTK toolbox, version 3.4.1. "HTK has been developed by the Machine Intelligence Laboratory (formerly know as the Speech Vision Robotics Group) at the Cambridge University Engineering Department (CUED) and Entropic Ltd. Microsoft has now licensed HTK back to CUED and is providing support so that CUED can redistribute HTK and provide development support via the HTK3 web site." (source: http://htk.eng.cam.ac.uk/) Notice that HTK is available for free download after registration and users must first agree to the license. Notice that the section 2.2 of the license terms mentions that HTK "either in whole or in part can not be distributed or sub-licensed to any third party in any form."

4.2 French

All French resources are *(c) Laboratoire Parole et Langage, Aix-en-Provence, France.*

4.2.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	passé, par, appris
b	b	voiced bilabial	beau, abris, baobab
t	t	voiceless alveolar	tout, thé, patte
d	d	voiced alveolar	doux, deux, addition
k	k	voiceless velar	cabas, psycho, quatre, kelvin
g	g	voiced velar	gain, guerre, second

Consonant Fricatives

SPPAS	IPA	Description	Examples
f	f	voiceless labiodental	fête, pharmacie
s	s	voiceless alveolar	sa, hausse, ce, garçon, option, scie
S	ʃ	voiceless postalveolar	choux, schème, shampooing
z	z	voiced alveolar	hasard, zéro, transit
Z	ʒ	voiced postalveolar	joue geai
v	v	voiced labiodental	vous, wagon, neuf heures

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	mou, homme
n	n	alveolar	nous, bonne
N	ŋ	voiced velar	camping, bingo

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	lit, ville, fil
R	ʁ	voiced uvular	roue, rhume, arrive

Semivowels

SPPAS	IPA	Description	Examples
j	j	palatal	payer, fille, travail
w	w	voiced labiovelar	oui, web, whisky
H	ɥ	labial-palatal	huit, Puy

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	crème, faite, peine, fête, maître, mètre, reine
a	a	open front unrounded	patte, là
a	ɑ	open back unrounded	pâte glas
9	œ	open-mid front rounded	sœur, neuf, œuf
o	ɔ	open-mid back rounded	sort, minimum
i	i	close front unrounded	si, île, régie, y
e	e	close-mid front unrounded	clé, les, chez, aller, pied, journée
o	o	close-mid back rounded	sot, hôtel, haut
u	u	close back rounded	cou, clown, roue
y	y	close front rounded	tu, sûr, rue
eu	ø	close-mid front rounded	ceux, jeûner, deux
eu	ə	schwa	le, reposer, monsieur, faisons

Notice that /eu/ is not an X-SAMPA symbol and is representing both /2/ and /@/. Notice also that /o/ is representing both /o/ and /O/.

Nasal vowels

SPPAS	IPA	Examples
a~	ɑ̃	sans, champ, vent, temps, Jean, taon
U~	ɛ̃	vin, pain, brin, printemps

SPPAS	IPA	Examples
U~	œ̃	un, parfum, brun
o~	ɔ̃	son, nom, bon

Notice that /o~/ should be /O~/: it will be corrected in a future version.

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech
fp	euh: filled pause

4.2.2 Pronunciation dictionary

The French pronunciation dictionary was created by Brigitte Bigi by collecting and merging several free dictionaries loaded from the web. Some pronunciations were added using the LIA_Phon tool. Many words were manually corrected and a large set of missing words and pronunciation variants were manually added.

It is distributed under the terms of the *GNU General Public License*.

4.2.3 Acoustic Model

The French acoustic model was created by Brigitte Bigi from various corpora mainly recorded at Laboratoire Parole et Langage. Special thanks are addressed to Roxane Bertrand, Béatrice Priego-Valverde, Sophie Herment, Amandine Michelas, Muriel Lalain and Christel Portes for kindly sharing their corpora.

It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

4.2.4 Syllabification configuration file

The syllabification configuration file corresponds to the one described in the paper (Bigi et al. 2010). It is distributed under the terms of the *GNU General Public License*.

4.3 Italian

4.3.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	primo, ampio, copertura
b	b	voiced bilabial	banca, cibo
t	t	voiceless alveolar	tranne, mito, Fiat
d	d	voiced alveolar	dove, idra
k	k	voiceless velar	cavolo, acuto, anche, quei
g	g	voiced velar	gatto, agro, glifo, ghetto

Consonant Fricatives

SPPAS	IPA	Description	Examples
f	f	voiceless labiodental	fatto, fosforo
s	s	voiceless alveolar	sano, scatola, presentire
S	ʃ	voiceless postalveolar	scena, sciame, pesci
z	z	voiced alveolar	sbavare, presentare, asma
v	v	voiced labiodental	vado, povero

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	mano, amare, campo
n	n	alveolar	nano, punto, pensare, anfibio
J	ɲ	palatal	gnocco, ogni
N	ŋ	voiced velar	fango, unghia, panchina, dunque

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	lato, lievemente
L	ʎ	palatal lateral	gli, glielo, maglia
r	r	alveolar trill	Roma, quattro, morte

Semivowels

SPPAS	IPA	Description	Examples
j	j	voiced palatal	ieri, più, Jesi
w	w	voiced labiovelar	uovo, fuoco, qui

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	elica, cioè
a	a	open front unrounded	alto, sarà
o	ɔ	open-mid back rounded	otto, posso, sarò
o	o	close-mid back rounded	ombra, come
e	e	close-mid front unrounded	vero, perché
i	i	close front unrounded	imposta, colibrì, zie
u	u	close back rounded	ultimo, caucciù, tuo

Affricates

SPPAS	IPA	Description	Examples
tS	t͡ʃ	voiceless postalveolar	Cennini, cinque, ciao
ts	t͡s	voiceless alveolar	sozzo canzone marzo
dz	d͡z	voiced alveolar	zaino zelare mezzo
dZ	d͡ʒ	voiced postalveolar	giungla, magia, fingere

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech
fp	eh, ah: filled pause

4.3.2 Pronunciation dictionary

The Italian dictionary was downloaded in 2011 from the Festival synthetizer tool. A large amount of the phonetization were manually corrected by Brigitte Bigi and a large set of missing words and pronunciation

variants were added manually.

It is distributed under the terms of the *GNU General Public License*.

4.3.3 Acoustic Model

The Italian acoustic model was created during the Evalita 2011 evaluation campaign, from the CLIPS Map-Task corpus (3h30), and updated during the Evalita 2014 evaluation campaign.

It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

4.3.4 Syllabification configuration file

The syllabification configuration file corresponds to the rules defined in the paper (Bigi and Petrone, 2014). It is distributed under the terms of the *GNU General Public License*.

4.4 Spanish

4.4.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	pozo, topo, perro
b	b	voiced bilabial	bestia, embuste, vaca
t	t	voiceless alveolar	tamiz, átomo
d	d	voiced alveolar	dedo, cuando, aldaba
k	k	voiceless velar	caña, laca, quise, kilo
g	g	voiced velar	gato, lengua, gatouerra

Consonant Fricatives

SPPAS	IPA	Description	Examples
b	β	voiced bilabial	bebé obtuso vivir curva
d	ð	voiced dental	dádiva, arder, anddmirar
g	ɣ	voiceless velar	trigo, amargo, sigue, signo
f	f	voiceless labiodental	fase, café
s	s	voiceless alveolar	saco, zapato, cientos, espita
z	z	voiced alveolar	isla, mismo, deshuesar

SPPAS	IPA	Description	Examples
S	ʃ	voiceless postalveolar	English, abacaxi, Shakira
T	θ	voiceless dental	cereal, encima, zorro, enzima, paz
x	x	voiceless velar	jamón, eje, reloj, general

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	madre, campo, convertir
n	n	alveolar	nido, anhelo, sin, álbum
J	ɲ	palatal	ñandú, cañón, enyesar
N	ŋ	voiced velar	cinco, venga, conquista

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	lino, alhaja, principal
L	ʎ	palatal lateral	llave, pollo, roughly
r	r	alveolar trill	rumbo, carro, amor
4	ɾ	alveolar flap	caro, bravo, eterno

Semivowels

SPPAS	IPA	Description	Examples
j	j	voiced palatal	aliada, cielo, amplio
w	w	voiced labiovelar	cuadro, fuego

Vowels

SPPAS	IPA	Description	Examples
a	a	open front unrounded	azahar
o	o	close-mid back rounded	boscoso
e	e	close-mid front unrounded	vehemente

SPPAS	IPA	Description	Examples
i	i	close front unrounded	dimitir, mío
u	u	close back rounded	cucurucho, dúo

Affricates

SPPAS	IPA	Description	Examples
tS	t͡ʃ	voiceless postalveolar	chubasco, acechar
ts	t͡s	voiceless alveolar	Ertzaintza, abertzale

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.4.2 Pronunciation Dictionary

The pronunciation dictionary was downloaded from the CMU web page in 2013. Brigitte Bigi converted the CMU phoneset to X-SAMPA, and changed the format of the file. It is distributed under the terms of the *GNU General Public License*.

4.4.3 Acoustic Model

The acoustic model was trained from Glissando corpus. We address special thanks to Juan-Maria Garrido for giving us access to this corpus. It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

GARRIDO, J. M. - ESCUDERO, D. - AGUILAR, L. -CARDEÑOSO, V. - RODERO, E. - DE-LA-MOTA, C. - GONZÁLEZ, C. - RUSTULLET, S. - LARREA, O. - LAPLAZA, Y. - VIZCAÍNO, F. - CABRERA, M. - BONAFONTE, A. (2013). *Glissando: a corpus for multidisciplinary prosodic studies in Spanish and Catalan*, Language Resources and Evaluation, DOI 10.1007/s10579-012-9213-0.

4.5 Catalan

4.5.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	pala
b	b	voiced bilabial	bala, via
t	t	voiceless alveolar	tela
d	d	voiced alveolar	donar
k	k	voiceless velar	cala
g	g	voiced velar	gala

Consonant Fricatives

SPPAS	IPA	Description	Examples
D	ð	voiced dental	cada
G	ɣ	voiceless velar	alga, mages
f	f	voiceless labiodental	fals
s	s	voiceless alveolar	si, sala
z	z	voiced alveolar	desde
S	ʃ	voiceless postalveolar	caixa
Z	ʒ	voiced postalveolar	mújol
v	v	voiced labiodental	va, vol
T	θ	voiceless dental	circus

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	mena
n	n	alveolar	nena
J	ɲ	palatal	any
N	ŋ	voiced velar	lingot, lingual

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	líquid
L	ʎ	palatal lateral	llamp
r	r	alveolar trill	carro
4	ɾ	alveolar flap	cara

Semivowels

SPPAS	IPA	Description	Examples
j	j	voiced palatal	iaia, naciós, iogurt
w	w	voiced labiovelar	veu, veuran

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	sec, veça
a	a	open front unrounded	sac
O	ɔ	open-mid back rounded	soc
o	o	close-mid back rounded	sóc
e	e	close-mid front unrounded	séc, cec
i	i	close front unrounded	sic, ric
u	u	close back rounded	suc
@	ə	schwa	contra, estada
U	ʊ	near-close near-back rounded	òpols

Affricates

SPPAS	IPA	Description	Examples
dZ	d͡ʒ	voiced postalveolar	metge
tS	t͡ʃ	voiceless postalveolar	cotxe

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.5.2 Pronunciation dictionary

The catalan pronunciation dictionary was downloaded in 2014 from the Ralf catalog of dictionaries for the Simon ASR system at http://spirit.blau.in/simon/import-pls-dictionary/. It was then converted (format and phoneset) by Brigitte Bigi. Some new words were also added and phonetized manually by Eva Bosch i Roura. New entries were then added from observed pronunciations in Glissando corpus.

It is distributed under the terms of the *GNU General Public License*.

4.5.3 Acoustic Model

The acoustic model was trained from Glissando corpus. We address special thanks to Juan-Maria Garrido for giving us access to this corpus. It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

> **GARRIDO, J. M. - ESCUDERO, D. - AGUILAR, L. -CARDEÑOSO, V. - RODERO, E. - DE-LA-MOTA, C. - GONZÁLEZ, C. - RUSTULLET, S. - LARREA, O. - LAPLAZA, Y. - VIZCAÍNO, F. - CABRERA, M. - BONAFONTE, A.** (2013). *Glissando: a corpus for multidisciplinary prosodic studies in Spanish and Catalan*, Language Resources and Evaluation, DOI 10.1007/s10579-012-9213-0.

4.6 English

4.6.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	pie, spy, cap
b	b	voiced bilabial	buy, cab
t	t	voiceless alveolar	tie, sty, cat, atom
d	d	voiced alveolar	dye, cad, do
k	k	voiceless velar	sky, crack, quick
g	g	voiced velar	guy, bag, luggage

Consonant Fricatives

SPPAS	IPA	Description	Examples
D	ð	voiced dental	thy, breathe, father
f	f	voiceless labiodental	phi, caff, fan
s	s	voiceless alveolar	sigh, mass
S	ʃ	voiceless postalveolar	shy, cash, emotion
z	z	voiced alveolar	zoo, has
Z	ʒ	voiced postalveolar	equation, pleasure, vision, beige
v	v	voiced labiodental	vie, have
T	θ	voiceless dental	thigh, math
h	h	voiceless glottal	high, ahead

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	my, smile, cam
n	n	alveolar	nigh, snide, can
N	ŋ	voiced velar	sang, sink, singer

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	lie, sly, gal
4	ɾ	alveolar flap	lyda, maddy, makita
r\	ɹ	alveolar approximant	red, try, very

Semivowels

SPPAS	IPA	Description	Examples
j	j	voiced palatal	yes, yacht, william
w	w	voiced labiovelar	wye, swine, why

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	dress, bed, fell, men
A	ɑ:	open back unrounded	palm, father, bra
A	ɒ	open back rounded	lot, pod, John
O:	ɔ:	open-mid back rounded	thought, Maud, dawn, fall
V	ʌ	open-mid back unrounded	strut, mud, dull, gun
i	i	close front unrounded	happy, serious
i:	i:	close front unrounded	fleece, seed, feel, sea
u:	u:	close back rounded	goose, food, chew, do
@	ə	schwa	a, baccus
I	ɪ	near-close near-front unrounded	kit, lid, fill, bin
U	ʊ	near-close near-back rounded	foot, full, woman
{	æ	near-open front unrounded	trap, pad, shall, ban

Affricates

SPPAS	IPA	Description	Examples
dZ	d͡ʒ	voiced postalveolar	giant, badge, jam
tS	t͡ʃ	voiceless postalveolar	China, catch

Others

SPPAS	IPA	Examples
aI	aɪ	price, ride, file, pie
aU	aʊ	mouth, loud, down, how
eI	eɪ	face, fail, vein, pay
OI	ɔɪ	choice, void, foil, boy
@U	oʊ	goat, code, foal, go
3:r	ɜ:r	liner, foundered, current

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.6.2 Pronunciation dictionary

The pronunciation dictionary is for North American English. It was downloaded in 2011 from the CMU web page. This Carnegie Mellon Pronouncing Dictionary (version 0.6) is Copyright (C) 1993-2008 by Carnegie Mellon University. We acknowledge CMU for distributing freely this resource and allowing its re-distribution.

Brigitte Bigi converted the original CMUdict encoded with ARPAbet into X-SAMPA and converted the format of the file in HTK-ASCII.

4.6.3 Acoustic Model

The acoustic model distributed in SPPAS resources was downloaded in 2014 from the VoxForge project at `http://www.voxforge.org/`.

Actually, two acoustic models are available:

- "eng" is context-dependent and does not contain the fillers;
- "eng-monophone" is context-independent, and contains the fillers (i.e. laugh, noise and un-transcribed speech). The latter was extracted from the first one.

Both English acoustic models are under the terms of the "GNU Public License".

It has to be noticed that the first one will time-align with a better accuracy but it can't be used if the speech to align contains fillers.

4.7 Polish

4.7.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	pas
b	b	voiced bilabial	
t	t	voiceless alveolar	
d	d	voiced alveolar	
k	k	voiceless velar	

SPPAS	IPA	Description	Examples
g	g	voiced velar	
c	c	voiceless palatal	

Consonant Fricatives

SPPAS	IPA	Description	Examples
f	f	voiceless labiodental	fac, feedback
s	s	voiceless alveolar	sabat, subaru
s\	ɕ	voiceless alveolo-palatal	świerszcz, śam
s‘	ʂ	voiceless alveolar with retroflex hook	
S	ʃ	voiceless postalveolar	
z	z	voiced alveolar	
z\	ʑ	voiced alveolo-palatal	źrebak
Z	ʒ	voiced postalveolar	
v	v	voiced labiodental	
x	x	voiceless velar	hak, chór
x	ç		
x	h		

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	
n	n	alveolar	
N	ŋ	voiced velar	

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	
r	r	alveolar trill	krok

Semivowels

SPPAS	IPA	Description	Examples
j	j	palatal	jak
w	w	voiced labiovelar	

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	test, ten
a	a	open front unrounded	pat, ptak
O	ɔ	open-mid back rounded	pot, kot
i	i	close front unrounded	pit, bit, miś
ɨ	ɨ	near-close central unrounded	ryba
u	u	close back rounded	bum
y	y	close front rounded	mysz

Nasal vowels

SPPAS	IPA	Examples
E~	ɛ̃	węże
o~	ɔ̃	wąż

Affricates

SPPAS	IPA	Description	Examples
tS	t͡ʃ	voiceless postalveolar	
ts	t͡s	voiceless alveolar	co
dz	d͡z	voiced alveolar	dzwon
dZ	d͡ʒ	voiced postalveolar	

4.7.2 Pronunciation Dictionary

The Polish pronunciation dictionary was downloaded in 2015 from the Ralf catalog of dictionaries for the Simon ASR system at http://spirit.blau.in/simon/import-pls-dictionary/.

It was then converted (format and phoneset) and corrected by Brigitte Bigi, thanks to the help of Katarzyna Klessa http://katarzyna.klessa.pl/. An update was done in 2017 to correct systematic errors.

It is distributed under the terms of the *GNU General Public License*.

4.7.3 Acoustic Model

The acoustic model was created by Brigitte Bigi. We address special thanks to Katarzyna Klessa for giving us access to a corpus.

It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

4.8 Portuguese

List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	pacto
b	b	voiced bilabial	bato
t	t	voiceless alveolar	tacto
d	d	voiced alveolar	dato
k	k	voiceless velar	cacto
g	g	voiced velar	gato

Consonant Fricatives

SPPAS	IPA	Description	Examples
f	f	voiceless labiodental	facto
s	s	voiceless alveolar	saca
S	ʃ	voiceless postalveolar	chato
z	z	voiced alveolar	zaca
Z	ʒ	voiced postalveolar	jacto
v	v	voiced labiodental	vaca
x	x	voiceless velar	rabão

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	mato
n	n	alveolar	nato
N	ŋ	voiced velar	hong-kong
J	ɲ	palatal	pinha

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	galo
L	ʎ	palatal lateral	galho
r	r	alveolar trill	pira
R	ʁ	voiced uvular	rato

Semivowels

SPPAS	IPA	Description	Examples
j	j	voiced palatal	yoga
w	w	voiced labiovelar	uísque

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	ego, eira
a	a	open front unrounded	parto
O	ɔ	open-mid back rounded	pôde
i	i	close front unrounded	hidra
e	e	close-mid front unrounded	pega, elo
o	o	close-mid back rounded	bola
u	u	close back rounded	hotel
y	y	close front rounded	emile
I	ɪ	near-close near-front unrounded	dois
U	ʊ	near-close near-back rounded	ido

Nasal vowels

SPPAS	IPA	Examples
a~	ɑ̃	anis
u~	ũ	humberto, unha

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.8.1 Pronunciation Dictionary

The Portuguese pronunciation dictionary was downloaded from the Ralf catalog of dictionaries for the Simon ASR system at http://spirit.blau.in/simon/import-pls-dictionary/. It was then converted (format and phoneset) and corrected by Brigitte Bigi.

It is distributed under the terms of the *GNU General Public License*.

4.8.2 Acoustic Model

The acoustic model was *NOT* trained from data. Monophones of other models were cut and pasted to create this one, mainly from the Spanish and the French models.

Portuguese data are welcome! Because data implies a better acoustic model then better alignments…

It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

4.9 Mandarin Chinese

4.9.1 List of phonemes

Any help is welcome to improve the quality of both Mandarin Chinese resources and of this documentation. Both are distributed without any warranty.

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	[illegible], [illegible]

SPPAS	IPA	Description	Examples
p_h	p^h	voiceless bilabial aspirated	□, □, □
t	t	voiceless alveolar	□, □
t_h	t^h	voiceless alveolar aspirated	□□
k	k	voiceless velar	□, □
k_h	k^h	voiceless velar aspirated	□, □

Consonant Fricatives

SPPAS	IPA	Description	Examples
f	f	voiceless labiodental	□, □, □
s	s	voiceless alveolar	□, □
s‘	ʂ	voiceless alveolar with retroflex hook	□ □
z‘	ʐ	voiced alveolar with retroflex hook	□, □
S	ʃ	voiceless postalveolar	□, □
x	x	voiceless velar	□, □
ss			□, □

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	□, □, □
n	n	alveolar	□, □, □
N	ŋ	voiced velar	□, □, □

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	□, □

Vowels

SPPAS	IPA	Description	Examples
a	a	open front unrounded	[illegible], [illegible], [illegible], [illegible], [illegible]
o	o	close-mid back rounded	[illegible], [illegible]
e	e	close-mid front unrounded	A, [illegible]
i	i	close front unrounded	[illegible], [illegible], [illegible] [illegible]
i_d	i̪	close front unrounded dental	[illegible], [illegible]
i‘	i˛	close front unrounded retroflex	[illegible][illegible], [illegible]
u	u	close back rounded	[illegible], [illegible], [illegible]
y	y	close front rounded	[illegible], [illegible], [illegible]
@‘	ə˛	schwa with retroflex hook	[illegible], [illegible]

Affricates

SPPAS	IPA	Description	Examples
ts	t͡s	voiceless alveolar	[illegible], [illegible]
tss			[illegible], [illegible]
ts_h	t͡sʰ	voiceless alveolar aspirated	[illegible], [illegible]
ts‘		voiceless alveolar retroflex hook	[illegible], [illegible]
ts_h‘			[illegible] [illegible]
ts_hs			[illegible], [illegible]

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.9.2 Pronunciation dictionary

The pronunciation dictionary was manually created for the syllables by Zhi Na. We address special thanks to her for sharing her work.

It is distributed under the terms of the *GNU General Public License*.

4.9.3 Acoustic model

The acoustic model was created by Brigitte Bigi from 2 corpora: the first one at Shanghai by Zhi Na, and another one by Hongwei Ding. We address special thanks to hers for giving us access to their corpus. Both recordings are a Chinese version of the Eurom1 corpus. See the following publication for details:

> **Daniel Hirst, Brigitte Bigi, Hyongsil Cho, Hongwei Ding, Sophie Herment, Ting Wang** (2013). *Building OMProDat: an open multilingual prosodic database*, Proceedings of Tools ans 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.

Notice that the current model was trained from a very small amount of data: this will impact on the results. Do not expect to get good performances for the automatic alignment.

More Mandarin Chinese data are welcome! Because more data implies a better acoustic model then better alignments…

The model is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

4.10 Southern Min (or Min Nan)

4.10.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description
b	b	voiced bilabial
d	d	voiced alveolar
p	p	voiceless bilabial
p_h	p^h	voiceless bilabial aspirated
t	t	voiceless alveolar
t_h	t^h	voiceless alveolar aspirated
k	k	voiceless velar
k_h	k^h	voiceless velar aspirated
g	g	voiced velar

Consonant Fricatives

SPPAS	IPA	Description
f	f	voiceless labiodental
s	s	voiceless alveolar

SPPAS	IPA	Description
S	ʃ	voiceless postalveolar
x	x	voiceless velar
ss		
v	v	voiced labiodental
z	z	voiced alveolar
h	h	voiceless glottal

Consonant Nasals

SPPAS	IPA	Description
m	m	bilabial
n	n	alveolar
N	ŋ	voiced velar

Consonant Liquids

SPPAS	IPA	Description
l	l	alveolar lateral

Semivowels

SPPAS	IPA	Description
w	w	voiced labiovelar

Vowels

SPPAS	IPA	Description
a	a	open front unrounded
o	o	close-mid back rounded
O	ɔ	open-mid back rounded
e	e	close-mid front unrounded
i	i	close front unrounded

SPPAS	IPA	Description
u	u	close back rounded
y	y	close front rounded
@	ə	schwa

Nasal vowels

SPPAS	IPA
a~	ɑ̃
e~	ɛ̃
o~	õ
O~	ɔ̃
u~	ũ
i~	ĩ

Affricates

SPPAS	IPA	Description
ts	t͡s	voiceless alveolar
ts_h	t͡sʰ	voiceless alveolar aspirated
dz	d͡z	voiced alveolar

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.10.2 Pronunciation Dictionary

The pronunciation dictionary was constructed from the most frequent observed pronunciations of the corpus described below.

It is distributed under the terms of the *GNU General Public License*.

4.10.3 Acoustic Model

It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

The acoustic model was trained from a corpus of spontaneous speech. We address special thanks to Sheng-Fu Wang for giving us access to his corpus.

> **S-F Wang, J. Fon** (2013). *A Taiwan Southern Min spontaneous speech corpus for discourse prosody*, Proceedings of Tools ans Resources for the Analysis of Speech Prosody, Aix-en-Provence, France, Eds B. Bigi and D. Hirst, ISBN: 978-2-7466-6443-2, pp. 20-23.

4.11 Cantonese

4.11.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description
p	p	voiceless bilabial
p_h	p^h	voiceless bilabial aspirated
t	t	voiceless alveolar
t_h	t^h	voiceless alveolar aspirated
k	k	voiceless velar
k_h	k^h	voiceless velar aspirated
k_w	k^w	voiceless velar labialized
k_h_w	k^{hw}	voiceless velar aspirated labialized

Consonant Fricatives

SPPAS	IPA	Description
f	f	voiceless labiodental
s	s	voiceless alveolar
S	ʃ	voiceless postalveolar
h	h	voiceless glottal

Consonant Nasals

SPPAS	IPA	Description
m	m	bilabial
n	n	alveolar
N	ŋ	voiced velar

Consonant Liquids

SPPAS	IPA	Description
l	l	alveolar lateral

Semivowels

SPPAS	IPA	Description
j	j	palatal
w	w	voiced labiovelar

Vowels

SPPAS	IPA	Description
E:	ɛː	open-mid front unrounded
a:	aː	open front unrounded
9:	œː	open-mid front rounded
O:	ɔː	open-mid back rounded
o	o	close-mid back rounded
e	e	close-mid front unrounded
8	ɵ	close-mid central rounded vowel
i:	iː	close front unrounded
u:	uː	close back rounded
y:	yː	close front rounded
6	ɐ	near-open central vowel
I	ɪ	near-close near-front unrounded
U	ʊ	near-close near-back rounded
@	ə	schwa

Affricates

SPPAS	IPA	Description
ts	t͡s	voiceless alveolar
ts_h	t͡sʰ	voiceless alveolar aspirated
tS	t͡ʃ	voiceless postalveolar
tS_h	t͡ʃʰ	voiceless postalveolar aspirated

4.11.2 Acoustic Model

The Cantonese acoustic model is copyrighted: *(C) DSP and Speech Technology Laboratory, Department of Electronic Engineering, the Chinese University of Hong Kong.*

This is a monophone Cantonese acoustic model, based on Jyutping of the Linguistic Society of Hong Kong (LSHK). Each state is trained with 32 Gaussian mixtures. The model is trained with HTK 3.4.1. The corpus for training is CUSENT, also developed in our laboratory.

Generally speaking, you may use the model for non-commercial, academic or personal use.

See COPYRIGHT for the details of the license: "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License".

We also have other well-trained Cantonese acoustic models. If you would like to use the models and/or the CUSENT corpus for commercial applications or development, please contact Professor Tan LEE for appropriate license terms.

The character pronunciation comes from Jyutping phrase box from the Linguistic Society of Hong Kong.

"The copyright of the Jyutping phrase box belongs to the Linguistic Society of Hong Kong. We would like to thank the Jyutping Group of the Linguistic Society of Hong Kong for permission to use the electronic file in our research and/or product development."

If you use this model for academic research, please cite:

> **Tan Lee, W.K. Lo, P.C. Ching, Helen Meng** (2002). *Spoken language resources for Cantonese speech processing*, Speech Communication, Volume 36, Issues 3–4, Pages 327-342

- Website: http://dsp.ee.cuhk.edu.hk
- Email: tanlee@ee.cuhk.edu.hk

4.12 Japanese

The linguistic resources for Japanese language are kindly shared by the authors of the Julius CSR engine.

For more detail, please contact csrc@astem.or.jp, or access the Julius website: http://julius.osdn.jp/en_index.php?q=index-en.html#about_models

4.13 Korean

Korean resources are under construction.

4.13.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
b	b	voiced bilabial	□, □□
p	p	voiceless bilabial	□
p_h	p^h	voiceless bilabial aspirated	□, □□
p_>	p̚	voiceless bilabial ejective	□
t	t	voiceless alveolar	□
t_h	t^h	voiceless alveolar aspirated	□
t_>	t̚	voiceless alveolar ejective	□, □
d	d	voiced alveolar	□, □, □
k	k	voiceless velar	□, □□□
k_h	k^h	voiceless velar aspirated	□
k_>	k̚	voiceless velar ejective	□, □□
g	g	voiced velar	□, □, □

Consonant Fricatives

SPPAS	IPA	Description	Examples
s	s	voiceless alveolar	□, □, □□
s_>	s̚	voiceless alveolar ejective	□

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	□, □ , □□
n	n	alveolar	□, □□
N	ŋ	voiced velar	□□, □□, □□

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	□, □, □
4	ɾ	alveolar flap	□, □□, □□

Semivowels

SPPAS	IPA	Description	Examples
j	j	voiced palatal	□, □, □, □, □
w	w	voiced labiovelar	□, □, □, □, □

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	□, □
A	ɑ	open back unrounded	□, □
i	i	close front unrounded	□, □
e	e	close-mid front unrounded	□, □
o	o	close-mid back rounded	□, □
u	u	close back rounded	□, □
2	ø	close-mid front rounded	□
V	ʌ	open-mid back unrounded	□, □
M	ɯ	close back unrounded	□, □

Affricates

SPPAS	IPA	Description	Examples
dz	d͡z	voiced alveolar	□
dZ	d͡ʒ	voiced postalveolar	□
tS_>	t͡ʃˀ	voiceless postalveolar ejective	□
tS_h	t͡ʃʰ	voiceless postalveolar aspirated	□

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.13.2 Pronunciation dictionary

The Korean pronunciation dictionary was manually created and is still under construction. **Any help is welcome!**

It is distributed under the terms of the *GNU General Public License*.

4.13.3 Acoustic Model

The acoustic model was *NOT* trained from data. Monophones of other models were cut and pasted to create this one, mainly the English and Taiwanese models.

Korean data are welcome! Because data implies a better acoustic model then better alignments…

It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

4.14 Naija

This work was financed by the French Agence Nationale pour la Recherche (ANR-16-CE27-0007), in the context of the NaijaSynCor project.

4.14.1 List of phonemes

Consonant Plosives

SPPAS	IPA	Description	Examples
p	p	voiceless bilabial	public, palaver
b	b	voiced bilabial	bye, bojuboju, boli
t	t	voiceless alveolar	two, tree, tranga
d	d	voiced alveolar	drop, duma, this
k	k	voiceless velar	ketu, cut, quick
g	g	voiced velar	gain, girl, guy

Consonant Fricatives

SPPAS	IPA	Description	Examples
f	f	voiceless labiodental	farm, phone, view
s	s	voiceless alveolar	centre, safe, zero
S	ʃ	voiceless postalveolar	cheque, sabi, ship
z	z	voiced alveolar	used, diesel, eze
v	v	voiced labiodental	visit, view
h	h	voiceless glottal	happy, hope, who
T	θ	voiceless dental	thing, ethnic, three

Consonant Nasals

SPPAS	IPA	Description	Examples
m	m	bilabial	make, milk, magaji
n	n	alveolar	knock, name, nitel
N	ŋ	voiced velar	bongo, sings

Consonant Liquids

SPPAS	IPA	Description	Examples
l	l	alveolar lateral	load, lokodan
r\	ɹ	alveolar approximant	radio, root, wrap

Semivowels

SPPAS	IPA	Description	Examples
j	j	voiced palatal	uni, yes, europe
w	w	voiced labiovelar	one, wait, wowo

Vowels

SPPAS	IPA	Description	Examples
E	ɛ	open-mid front unrounded	air, early, egg, men

SPPAS	IPA	Description	Examples
a	a	open front unrounded	our, ask, above
O	ɔ	open-mid back rounded	us, onion, all, oba
i	i	close front unrounded	each, even, ile, is
e	e	close-mid front unrounded	alone, eko
o	o	close-mid back rounded	obodo, ojo
u	u	close back rounded	ugu, una, upo

Nasal vowels

SPPAS	IPA	Examples
a~	ɑ̃	auntie, african, commander
e~	ẽ	fiyen, britain, town
E~	ɛ̃	calendar, men, accent
i~	ĩ	admin, ani, benin
O~	ɔ̃	election, lokodan, million
u~	ũ	remove, segun, broken

Affricates

SPPAS	IPA	Description
tS	t͡ʃ	voiceless postalveolar
dZ	d͡ʒ	voiced postalveolar

Others

SPPAS	IPA	Examples
aI	aɪ	I, write, type
aU	aʊ	out, town
OI	ɔɪ	oil, boy
eI	eɪ	a, eight, age

Fillers

SPPAS	Description
@@	laughter
gb	noises, unintelligible speech
dummy	un-transcribed speech

4.14.2 Pronunciation Dictionary

The dictionary was originally created by extracting the lexicon of the corpus published in annex of (Deuber 2005). New words with their orthographic variants and pronunciations were added to the dictionary by team of four transcribers, native speakers of the language.

It is distributed under the terms of the *GNU General Public License*.

4.14.3 Acoustic Model

The Naija acoustic model was created in 2017-07 by Brigitte Bigi with the SPPAS training scripts .

An initial model was created on the basis of other language prototypes. Such prototypes were mostly extracted from the English acoustic model. For the missing models of phonemes, the nasals /O~/, /a~/, /e~/, /i~/ and /u~/ were picked off Southern Min language, and /E~/ was extracted from French language using /U~/ prototype. The vowels /a/ and /e/ were extracted from the French model; and finally /O/ and /o/ from the Italian one. The fillers were also added to the model in order to be automatically time-aligned too: silence, noise, laughter.

The acoustic model was then trained with a set of 8 files (totalling 3 min 29 seconds in length.) manually phonetized and time-aligned.

It is distributed under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License*.

5

Analyses

5.1 Introduction

The analysis tools are useful for the analyses of annotated files: display the automatic alignments with the audio signal, estimates statistics on the annotations, filter the annotated data to get only the annotations you are interested in, etc.

To execute a specific analysis tool, select file(s) in the file explorer of the main frame, then click on the button of the tool. It will open the window of the tool, and add the file(s) in its file manager.

Six tools are available:

1. `DataRoamer` allows to explore the annotated files: cut/copy/paste/rename/duplicate tiers, move a tier from one file to another one, etc.
2. `AudioRoamer` allows to play and manage audio files: extract a channel, see clipping rates, change framerate, etc.
3. `IPUscriber` is useful to perform manual orthographic transcription.
4. `Visualizer` displays audio files and annotated files, and is very useful to take a screenshot! Easy way to zoom/scroll, change colours, choose the tiers to display, etc;
5. `DataFilter` allows to select a sub-set of annotations: fix a set of filters to create new tiers with only the annotations you are interested in!
6. `Statistics` estimates the number of occurrences, the duration, etc. of the annotations, and allows to save in CSV (for Excel, OpenOffice, R, MatLab,…).

All of such tools share the same look with:

- a menu at left
- a toolbar at top
- a list of files
- a notebook to open files in the tabs

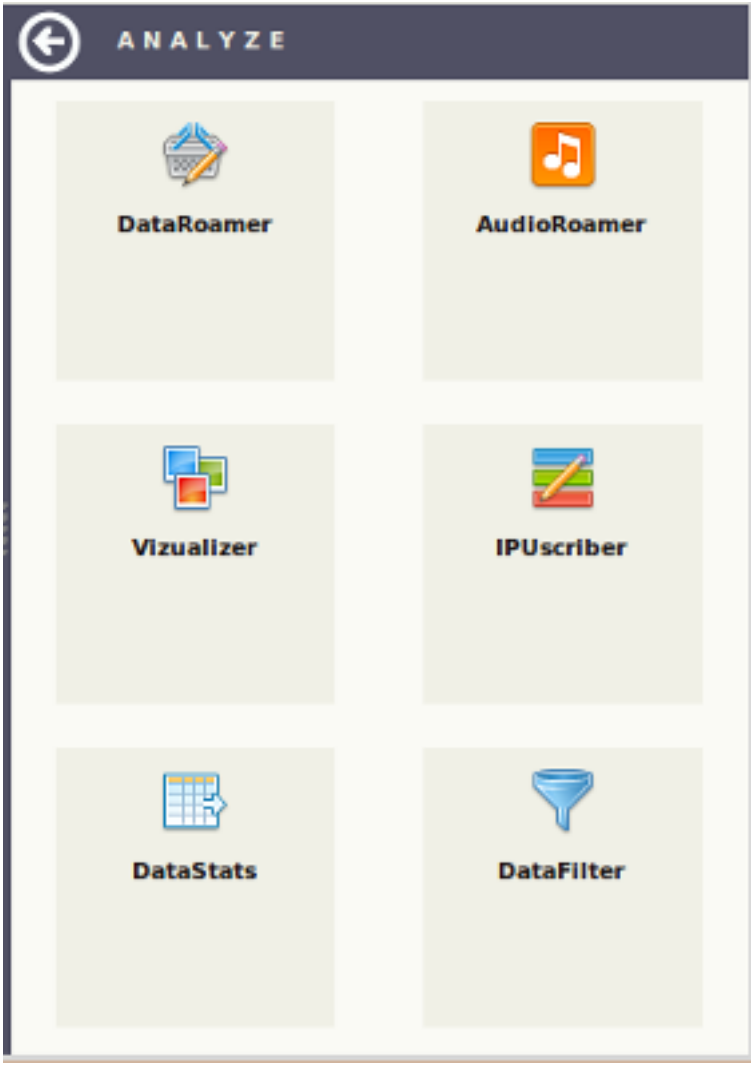


Figure 5.1: The analysis tools

5.2 DataRoamer

`DataRoamer` displays detailed information about annotated files and allows to manage the tiers: cut/copy/paste/rename/duplicate tiers, move a tier from one file to another one, etc.

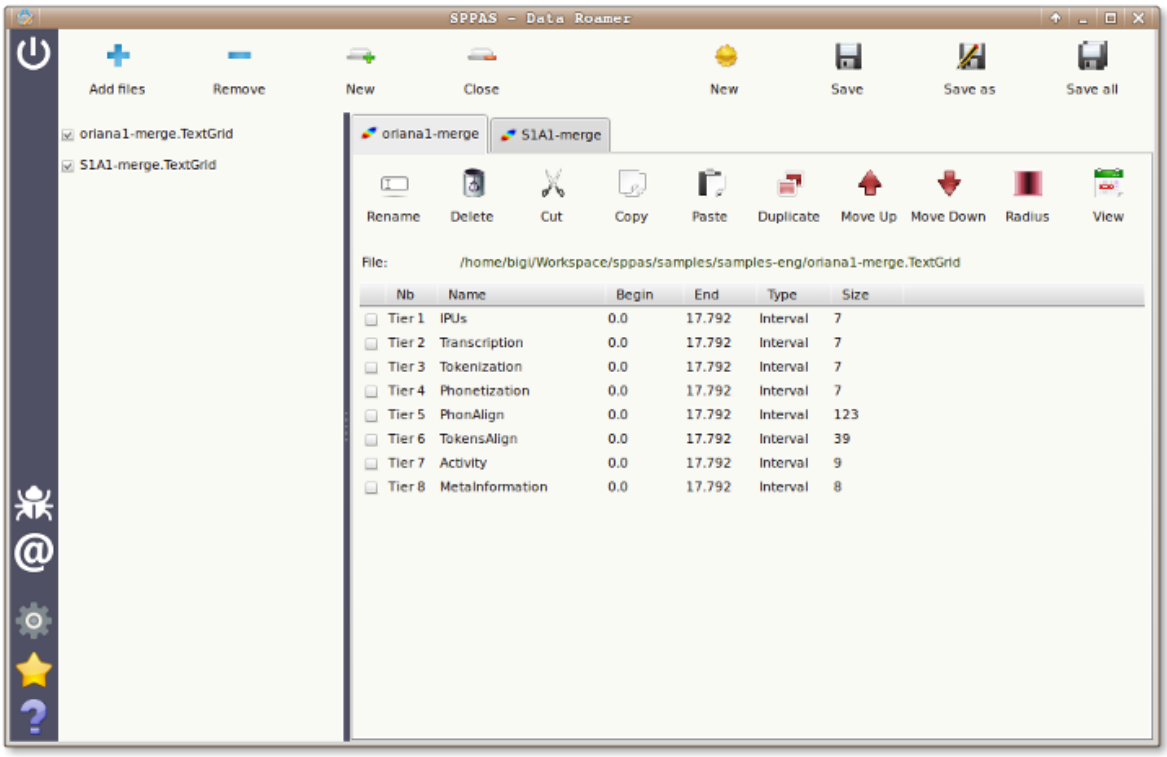


Figure 5.2: DataRoamer: explore annotated files

When an annotated file is checked in the list of files, the file is opened in a new panel of the current tab of the notebook and information about this file are displayed. At top, the name of the file is in dark green color. It is turned to blue color if some changes were applied. Below the name of the file, each tier is represented in a row of a spreadsheet:

- 1st column is a check box: select/deselect the tier;
- 2nd column indicates the number of the tier;
- 3rd column indicated the name of the tier;
- 4th column indicates the lower time value of the annotations;
- 5th column indicates the higher time values of the annotations;
- 6th column is the type of the tier: either “Interval” or “Point”;
- last column is the number of annotations in the tier.

The buttons of the toolbar can be applied on the tier(s) selected in the same tab of the notebook. For example, it is possible to copy a tier of a file and to paste it in another file only if they are both open in the same tab.

- Rename: Fix a new name to the selected tier. If the given name is already assigned to a tier in the same file, a number is automatically appended to the end of the new name.
- Delete: Definitively delete the selected tier. To recover the tier, the only way is to close the file without saving changes and to re-open it.

- Cut: Delete the tier of the file and put it in the dashboard.
- Copy: Copy the selected tier into the dashboard.
- Paste: Paste the content of the dashboard into the currently selected file.
- Duplicate: Duplicate the selected tier in the file. A number is automatically appended to the end of name of the copy.
- Move Up: Move the selected tier up in the list of tiers of the file.
- Move Down: Move the selected tier down in the list of tiers of the file.
- Radius: Fix the radius value of each localization of each annotation. A radius value indicates the vagueness around the point. It is commonly ranging from 1ms for very (very) precise annotations up to 40ms for annotations of videos for example. This value is safely saved only in XRA format. Other formats don't consider vagueness.
- View: Open a window and display the annotations of the tier.

Only the `Delete` and `Radius` actions can be applied on several selected tiers at a time. The other actions can be applied only on one selected tier.

5.3 AudioRoamer

`AudioRoamer` allows to play audio files, to display general information about a digitalized audio-PCM file and to manipulate the file.

Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. In a PCM stream, the amplitude of the analog signal is sampled regularly at uniform intervals. A PCM stream has two basic properties that determine the stream's fidelity to the original analog signal:

1. the sampling rate, which is the number of times per second that samples are taken;
2. the bit depth, which determines the number of possible digital values that can be used to represent each sample.

Common sampling frequencies are 48 kHz as used with DVD format videos, or 44.1 kHz that was used in Compact discs. Common sample depths is 16 bits per sample: it allows values to range from -32768 to 32767.

Support for multichannel audio depends on file format and relies on interweaving or synchronization of streams.

When an audio file is checked in the list of files, a new page is opened in the notebook. The main information about the audio file are then displayed in the panel at the middle and a player is displayed at bottom. The audio file is not loaded in memory, so event very long audio files can be displayed. At the top of the page, a button "Want more?" can be clicked: the audio file will then be fully read and analyzed, then a new window will be opened to display detailed information. It will also allow to change some properties of the audio file, extract a channel, etc.

5.3.1 Properties of the audio file

The main properties of any audio file are displayed. SPPAS can open audio files of type "Waveform Audio" file format (WAVE), a Microsoft and IBM audio file format standard and "SunAu" file format, an audio file format introduced by Sun Microsystems. The following information are extracted or deducted of the header of the audio file:

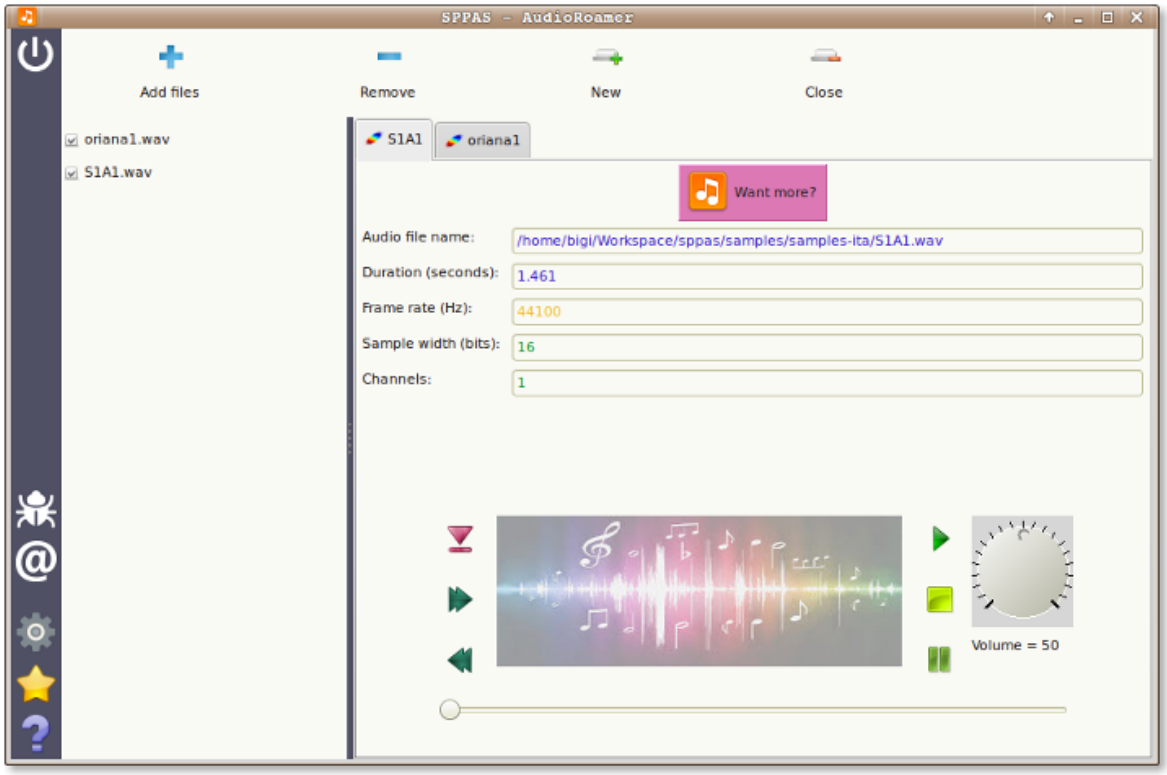


Figure 5.3: AudioRoamer: play and manage audio files

- the duration given in seconds with a maximum of 3 digits;
- the frame rate given in Hz;
- the sample width in bits (one of 8, 16, 24 or 32);
- the number of channels.

A color code is used to indicate if the file is compatible with SPPAS automatic annotations:

- Green color: the value is perfectly matching the expectations;
- Orange color: the value is not the expected one but automatic annotations will be able to convert it to the right one;
- Red color: none of the automatic annotations can work with the given value. The file must be modified.

5.3.2 Playing audio files

The following shortcuts can be used:

- TAB: Play
- ESC: Stop
- F6: Rewind
- F7: Pause
- F8: Next

5.3.3 Want more?

`AudioRoamer` allows to display a large amount of information for each channel. For a long audio file, it can take a while to estimate such information… For example, a mono-audio file of 243 seconds (21.5Mb) is loaded in 20 seconds. **So be patient! It's worth it!**

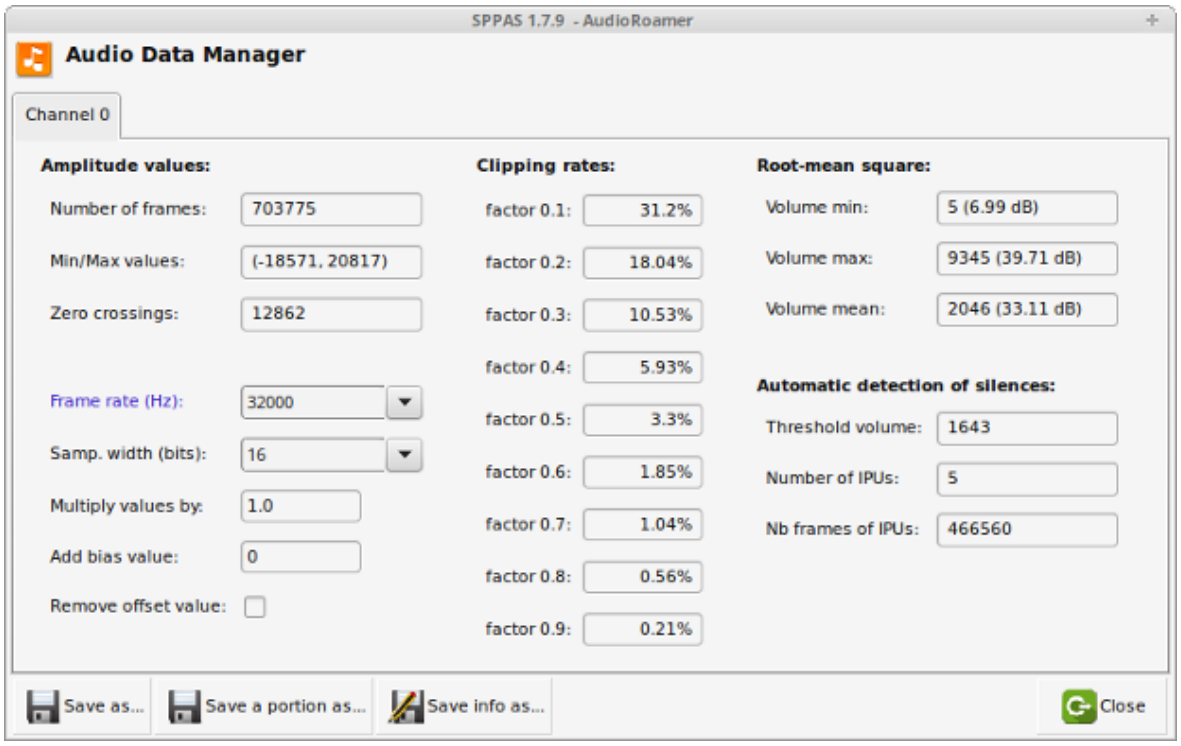


Figure 5.4: AudioRoamer if you want more

There are 3 main columns of information, related to amplitude, clipping and volume.

At the bottom of the window, it is possible to click on buttons to:

1. Save the channel currently displayed;
2. Save a portion (from…to…) of the channel;
3. Save the displayed information in a text file;
4. Close the window.

Amplitude

The amplitude is a variable characterizing the sinusoidal oscillation of the digital-audio recording. It gives the deflection of a physical quantity from its neutral position (zero point) up to a positive or negative value. With sound waves, it is the extent to which air particles are displaced, and this amplitude of sound or sound amplitude is experienced as the loudness of sound. The loudness perception of a sound is determined by the amplitude of the sound waves – the higher the amplitude, the louder the sound or the noise.

The first column of `AudioRoamer` Data Manager gives amplitude values of each channel of the PCM file:

- Number of frames: the number of samples, i.e. the number of amplitude values;
- Min/Max values: minimum and maximum amplitude values in the channel;

- Zero crossings: number of times continuous values are crossing the 0 value.

The 5 information below can be modified and modifications can be saved separately for each channel:

- Frame rate: the sampling frequency observed in the channel is included in the default list of possible values. If it is modified, the color is changed.
- Samp. width: the bit depth observed in the channel is included in the list of possible values. If it is modified, the color is changed.
- Multiply values by: all samples in the original channel are multiplied by the floating-point value factor. Sample values are truncated in case of overflow.
- Add bias value: a bias value is added to each sample. Sample values wrap around in case of overflow.
- Remove offset value: the average over all values is deduced to each sample. Sample values wrap around in case of overflow.

Clipping

The second column displays the clipping rates of the sample values given a factor. It will consider that all values outside the interval are clipped. The clipping rate is given as a percentage related to the total number of values. For example, if factor is 0.5 and clipping rate is 10%, it means that 10% of the amplitude values are more than the half of the maximum amplitude. This value is generally an expected rate while preparing audio recordings.

Volume

RMS: Volume is estimated with root-mean-square (RMS) of the samples, i.e. sqrt(sum(S_i^2)/n). This is a measure of the power in an audio signal. Between parenthesis, the volume in dB is estimated as: 20.log10(rms). Doubling of the rms value leads to an increase of 6.02dB.

Automatic detection of silences: Finally, the result of an automatic detection of silences is given. This information is given for information only: It is estimated with default parameters which should be adapted!

5.4 IPUscriber

`IPUscriber` is useful to perform manual orthographic transcription. When an audio file is checked in the list of files, the program search for a file with the IPUs Segmentation, i.e. a file with the same name (in the same directory) with a known extension like .xra, .TextGrid, etc. If this file is not found, an error message is displayed. Otherwise, the audio file is opened and the annotated file is loaded in a new page of the notebook.

Silences are hidden and the IPUs are displayed in a list of boxes with:

- at left: the name of the tier followed by the number of the interval, then the time values of start and end of the interval, and the duration of the interval between parenthesis.
- at right: the content text of the IPU.

At the bottom of the tab, green buttons perform actions on the audio file: get information, play, auto-play, make pause and stop playing. The following keyboard shortcuts can also be used:

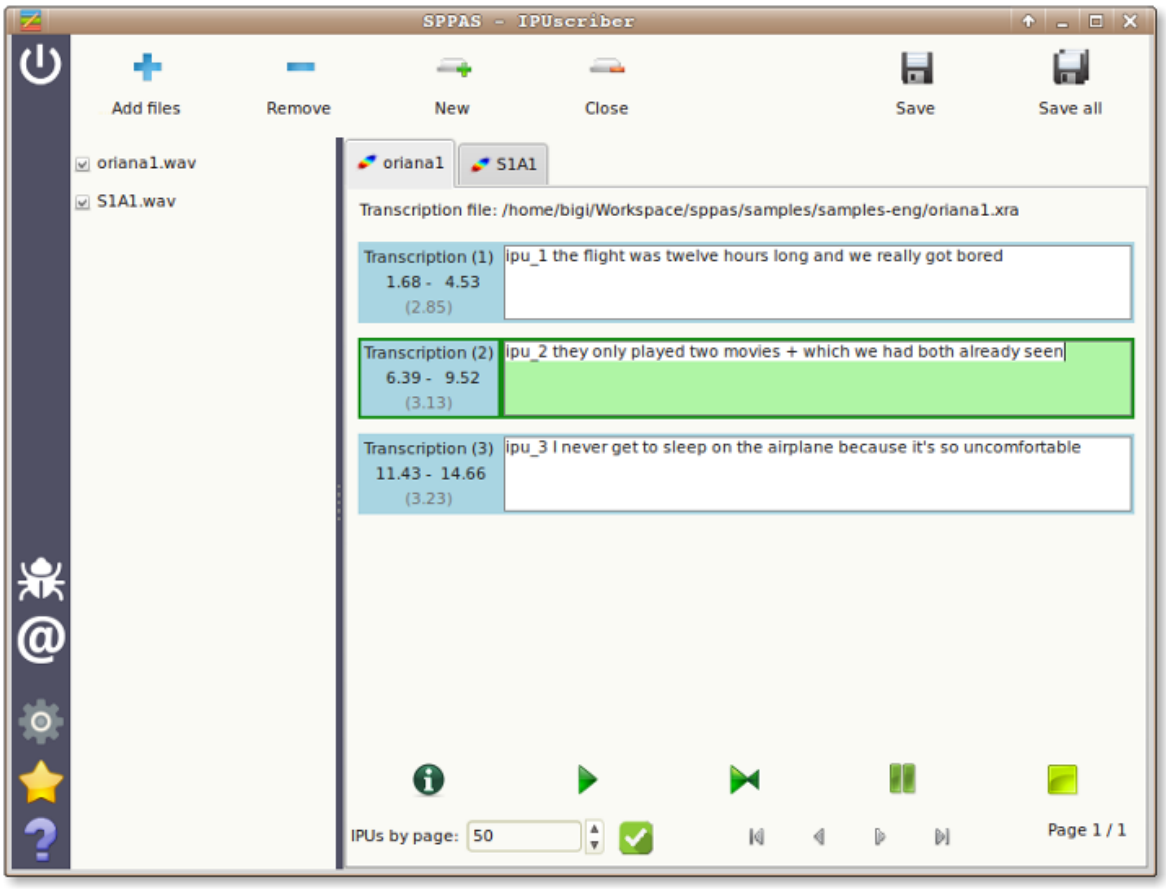


Figure 5.5: IPUscriber: for manual orthographic transcription

- TAB: Play
- ESC: Stop
- F6: Rewind
- F7: Pause
- F8: Next

By default, only 50 IPUs are displayed at a time in a tab. To get access to the next/previous IPUs, 4 small buttons in gray allow to navigate in the pages.

To transcribe an IPU, click on the IPU text box, play sound and write the corresponding text: refer to the transcription convention of this document.

5.5 Visualizer

`Visualizer` displays speech files and annotated files, and is very useful to take a nice screenshot.

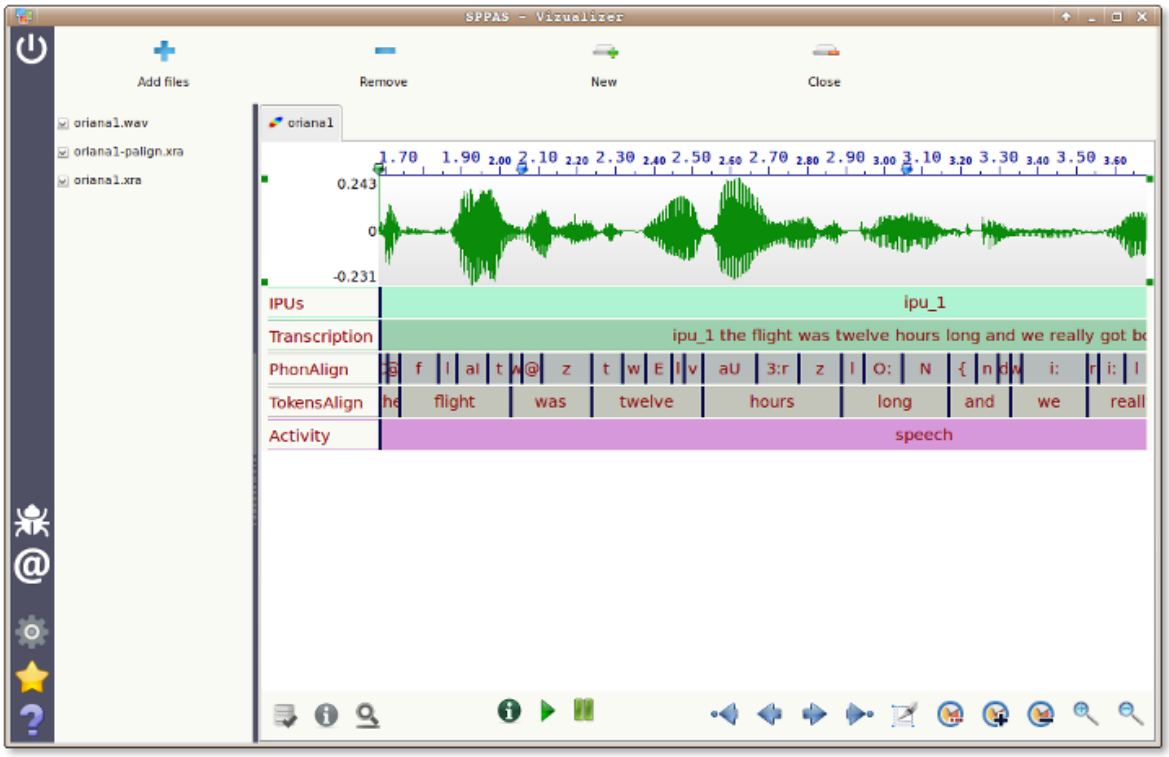


Figure 5.6: Visualizer: displays audio files and annotated files

This tool is still under-development, some “troubles/crashes” can occur while using it… however the data will never been corrupted!

When a file is checked on the file of files, it is opened and displayed in the currently opened tab of the notebook. There is not limit on the number of audio files that can be displayed nor on the number of annotated files. To select a displayed file, click anywhere of the displayed part.

At the bottom of the tab, 3 series of buttons are available:

- at left, red buttons allow to manage the selected annotated file:

1. select to tiers to display
2. get information about the file and tiers
3. search for annotations in the tier

- at the middle, green buttons are related to an audio file

1. get information about the file
2. play sound
3. pause

- at right, blue buttons are related to zoom and scroll.

1. move at the beginning of the file
2. move at left (show previous data)
3. move at right (show next data)
4. move at the end of the file
5. show the selection. Selected part is the one between the blue arrows of the rule
6. display a specific time period (from… to…)
7. zoom in: show a smaller period
8. zoom out: show a higher period
9. vertical zoom in
10. vertical zoom out

5.6 DataFilter

`DataFilter` allows to select annotations: define a set of filters to create new tiers with only the annotations you are interested in! This system is based on the creation of 2 different types of filters:

1. single filters, i.e. search in a/several tier(s) depending on the data content, the time values or the duration of each annotation;
2. relation filters, i.e. search on annotations of a/several tier(s) in time-relation with annotations of another one.

These later are applied on tiers of many kind of input files (TextGrid, eaf, trs, csv…). The filtering process results in a new tier, that can re-filtered and so on.

5.6.1 Filtering annotations of a tier: SingleFilter

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.

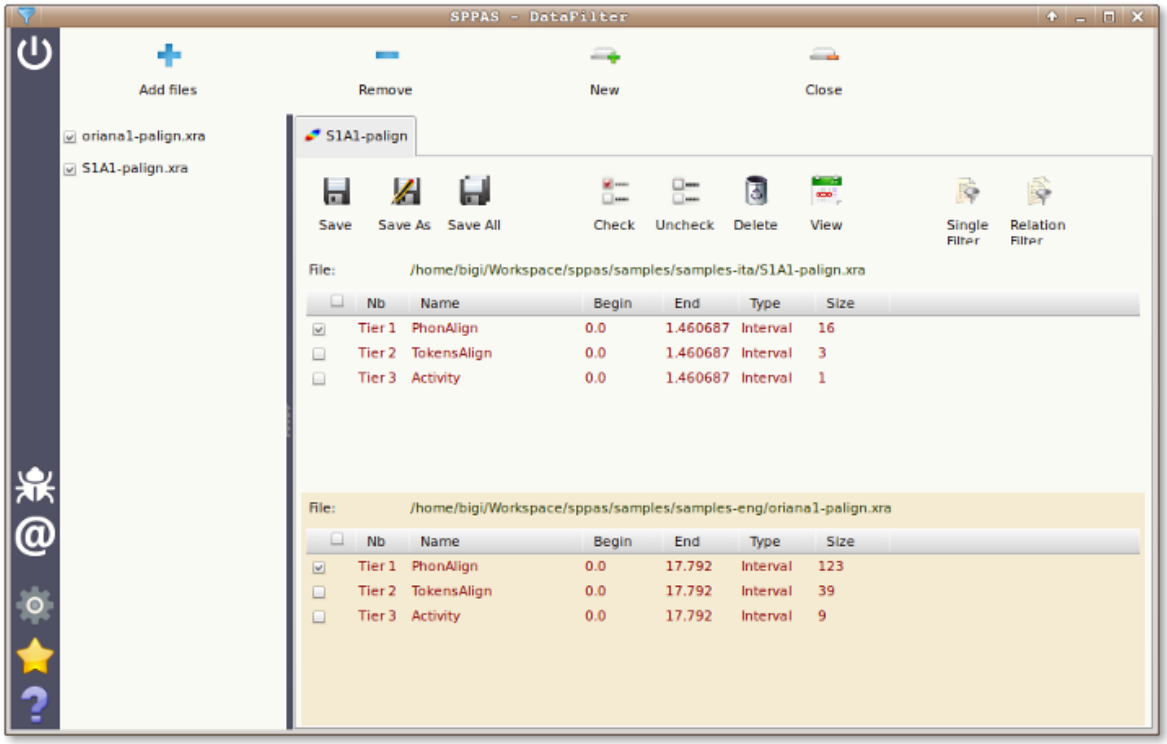


Figure 5.7: DataFilter: select annotations

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.

All the given filters are then summarized in the “SingleFilter” frame. 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.

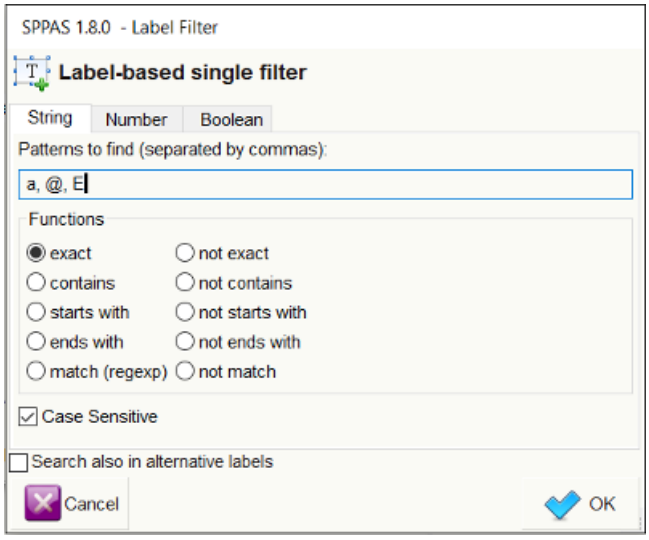


Figure 5.8: Frame to create a filter on annotation labels. In that case, filtering annotations that exactly match either a, @ or E

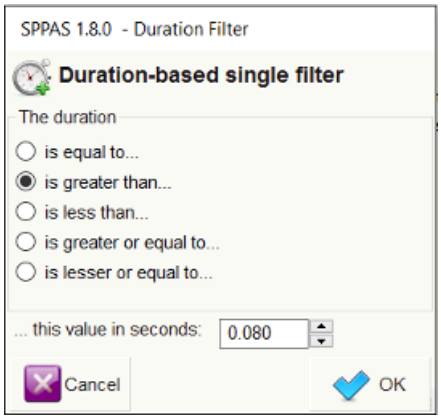


Figure 5.9: Frame to create a filter on annotation durations. In that case, filtering annotations that are during more that 80 ms

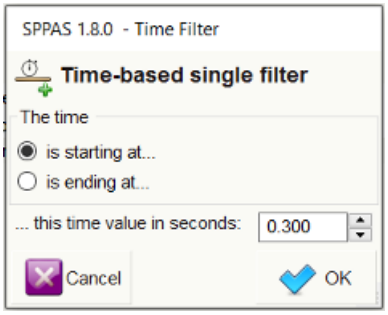


Figure 5.10: Frame to create a filter on annotation time values. In that case, filtering annotations that are starting after the 5th minute.

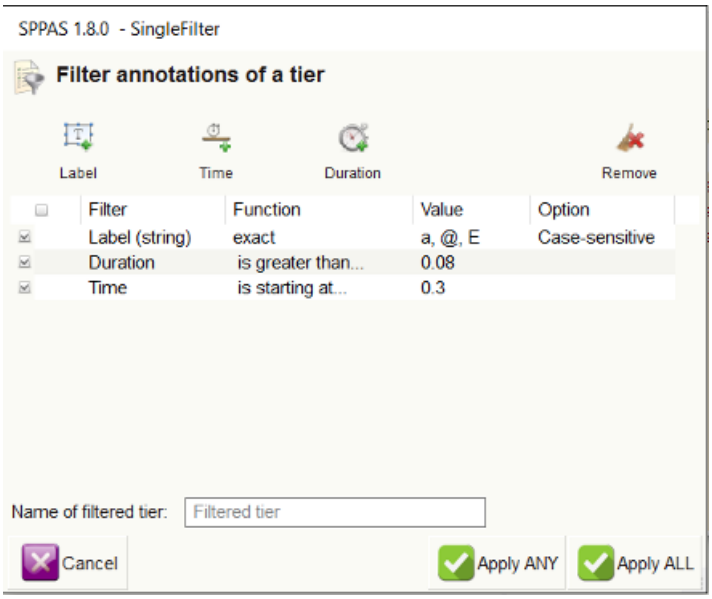


Figure 5.11: DataFilter: SingleFilter frame

5.6.2 Filtering on time-relations between two tiers

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).

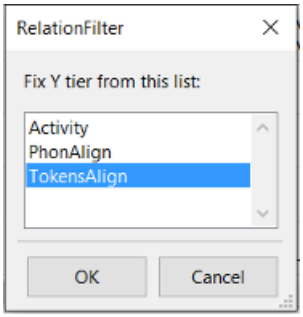


Figure 5.12: Fix time-relation tier name

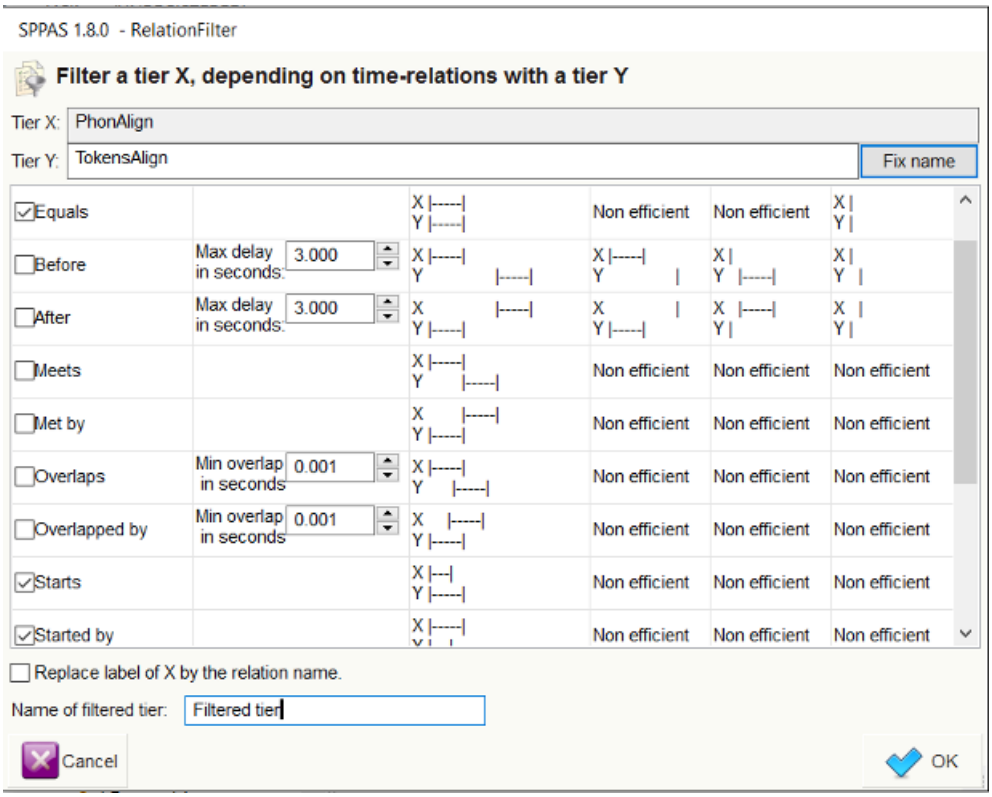


Figure 5.13: DataFilter: RelationFilter frame

5.7 Statistics

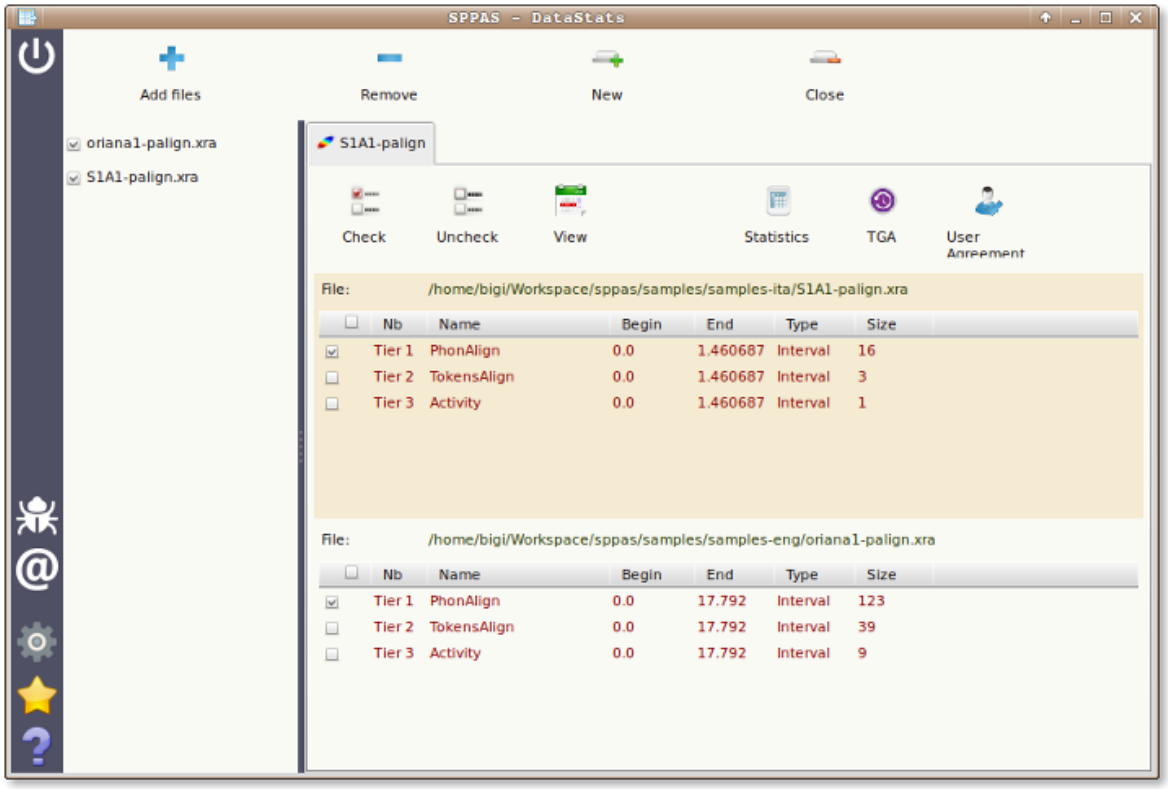


Figure 5.14: Statistics: descriptive statistics and TGA

5.7.1 Descriptive statistics

It allows to estimate the number of occurrences, the duration, etc. of the annotations of a set of selected tiers, and allows to save 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:

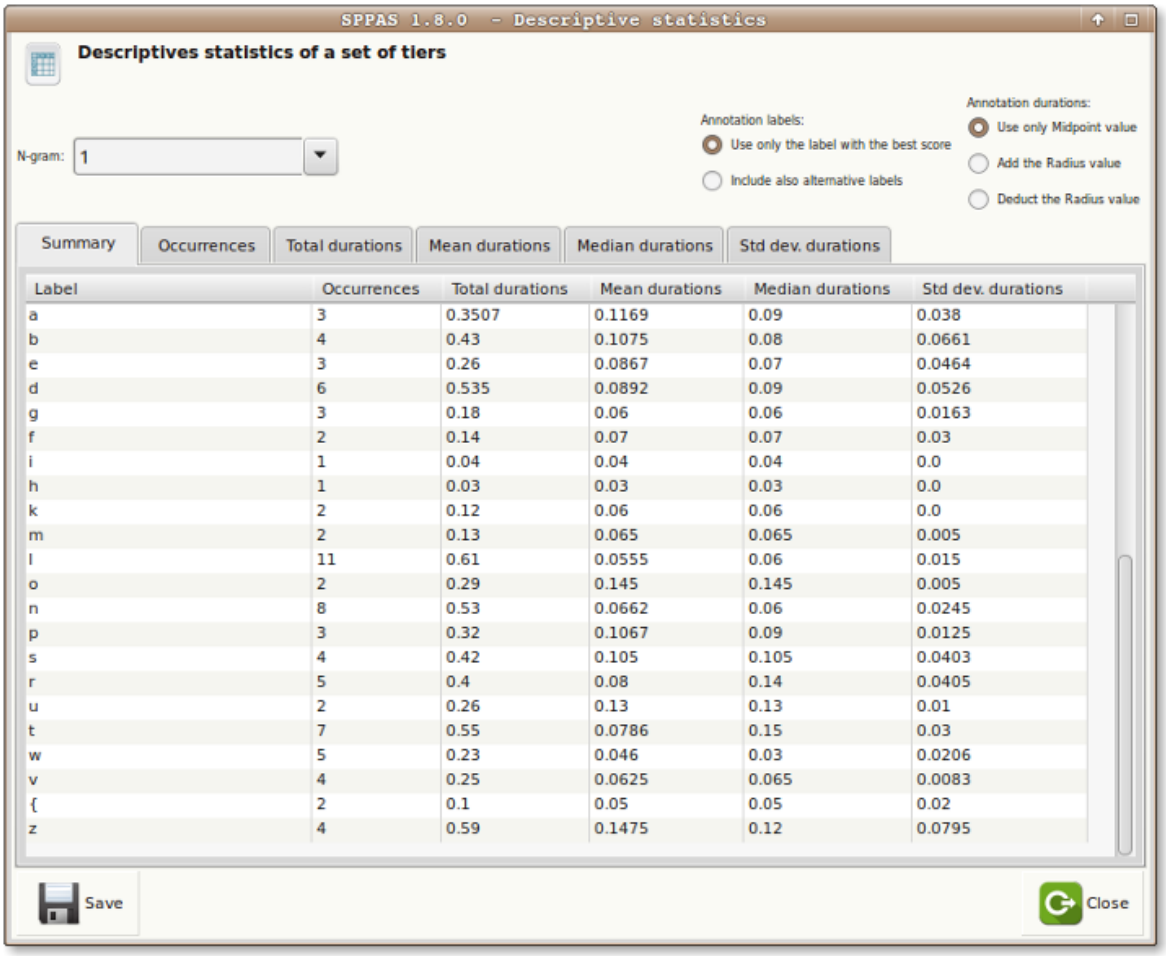
- occurrences: the number of observations
- total durations: the sum of the durations
- mean durations: the arithmetic mean of the duration
- median durations: the median value of the distribution of durations
- std dev. durations: the standard deviation value of the distribution of durations

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 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).



Label	Occurrences	Total durations	Mean durations	Median durations	Std dev. durations
a	3	0.3507	0.1169	0.09	0.038
b	4	0.43	0.1075	0.08	0.0661
e	3	0.26	0.0867	0.07	0.0464
d	6	0.535	0.0892	0.09	0.0526
g	3	0.18	0.06	0.06	0.0163
f	2	0.14	0.07	0.07	0.03
i	1	0.04	0.04	0.04	0.0
h	1	0.03	0.03	0.03	0.0
k	2	0.12	0.06	0.06	0.0
m	2	0.13	0.065	0.065	0.005
l	11	0.61	0.0555	0.06	0.015
o	2	0.29	0.145	0.145	0.005
n	8	0.53	0.0662	0.06	0.0245
p	3	0.32	0.1067	0.09	0.0125
s	4	0.42	0.105	0.105	0.0403
r	5	0.4	0.08	0.14	0.0405
u	2	0.26	0.13	0.13	0.01
t	7	0.55	0.0786	0.15	0.03
w	5	0.23	0.046	0.03	0.0206
v	4	0.25	0.0625	0.065	0.0083
{	2	0.1	0.05	0.05	0.02
z	4	0.59	0.1475	0.12	0.0795

Figure 5.15: Descriptive statistics

5.7.2 TGA - Time Group Analyzer

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.

The 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 visualisations 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.

> *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 modelling of speech timing.

So, what's the novelty…

TGA is partly implemented in SPPAS. The Statistics analyses tool of SPPAS allows to estimates TGA within the SPPAS framework. It results in the following advantages:

- it can read either TextGrid, csv, Elan, HTK, or Sclite, or any file format supported by SPPAS,
- it can save TGA results either as a table in a CSV file or as an annotation file (of any of the format supported by SPPAS),
- it estimates 2 linear regressions (the y-axis is the duration in both cases):
    1. with x-axis based on positions, like in the inline TGA
    2. with x-axis based on time-stamps, like in the AnnotationPro+TGA

5.7.3 User agreement

SPPAS implements the estimation of the Cohen's Kappa. It is currently limited to the evaluation of this user agreement between labels of 2 tiers with the same number of intervals. It is under-development…

6

Scripting with Python and SPPAS

6.1 Introduction

SPPAS implements an Application Programming Interface (API), named *annotationdata*, to deal with annotated files.

annotationdata 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…). *annotationdata* is based on the Programming Language Python 2.7. It won't work with Python 3.x. Notice that this API will be fully re-implemented in the scope of 1/ compatibility with Python > 3.3, 2/ PEP8 and PEP257 compliance and 3/ internationalization of the message. The new version will available in the middle of 2018 but the current one will be maintained until the end of 2018 or later.

In this chapter, it is assumed that Python 2.7 is already 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 *annotationdata* 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 `Transcription` 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 programs are included in
the package directory *documentation*, then in the folder *solutions*.
```

6.2 A gentle introduction to programming

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 “>>>”.

Figure 6.1: The Python IDLE logo

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"
>>>cc = u"ç"
>>>hello = "Hello world!"
>>>vrai = True
```

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

MISSING:Variable declarations and print in the Python IDLE./etc/screenshots/python_idle_example.png

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
>>> 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 use and/or 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'>
```

The following 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 changed. This is called a "cast", like in the following:

```
>>> 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']
>>> to_buy = fruits[1:2]
>>> to_buy
['tomatoes']
```

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. If the result of the condition is true and/or false, then a given action occurs. Statements of conditions are written with operators, as equal, less than, greater than, etc. 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 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 do not 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: ~~~~~~ {.python} >>> for key in ['apples', 'peers', 'tomatoes', 'babanas']: … 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.

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 programs.

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!")
```

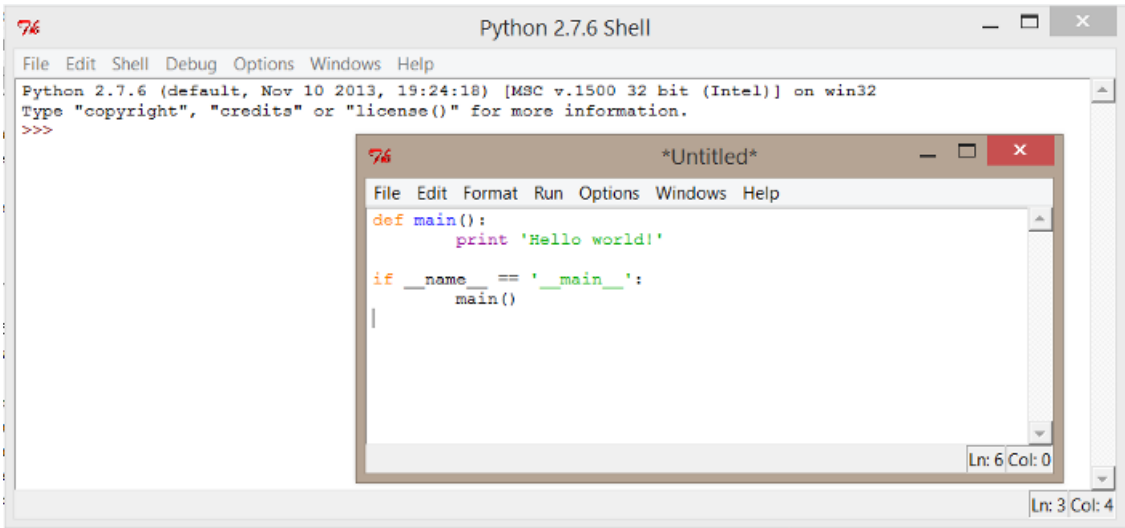


Figure 6.2: Hello world! in a Python script

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:

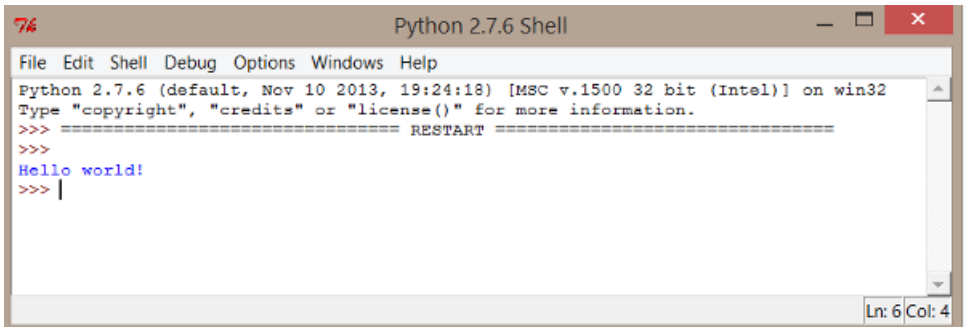


Figure 6.3: Output of the first script

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 Functions

Simple function

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

Here is an example of function:

```
def print_vowels():
    """ Print the list of French vowels on the screen. """

    vowels = ['a', 'e', 'E', 'i', 'o', 'u', 'y', '@', '2', '9', 'a~', 'o~', 'U~']
    print("List of French vowels:")
    for v in vowels:
        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 including 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 new version is as follow:

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

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

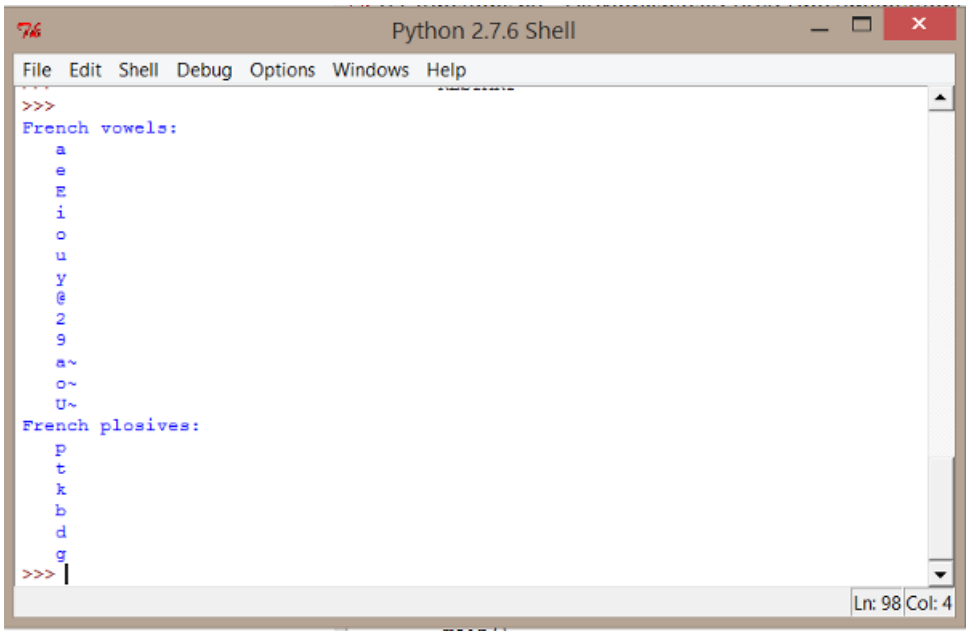


Figure 6.4: Output of the second script

Practice: add 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! Lets update, or refactor, our printing function to *make it more generic*.

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. 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:

```
27    def print_list(mylist, message=""):
28        """ Print a list on the screen.
29
30        :param mylist: (list) is the list to print
31        :param message: (string) is an optional message to print before each element
32
33        """
34        for item in mylist:
35            print("{0} {1}".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)

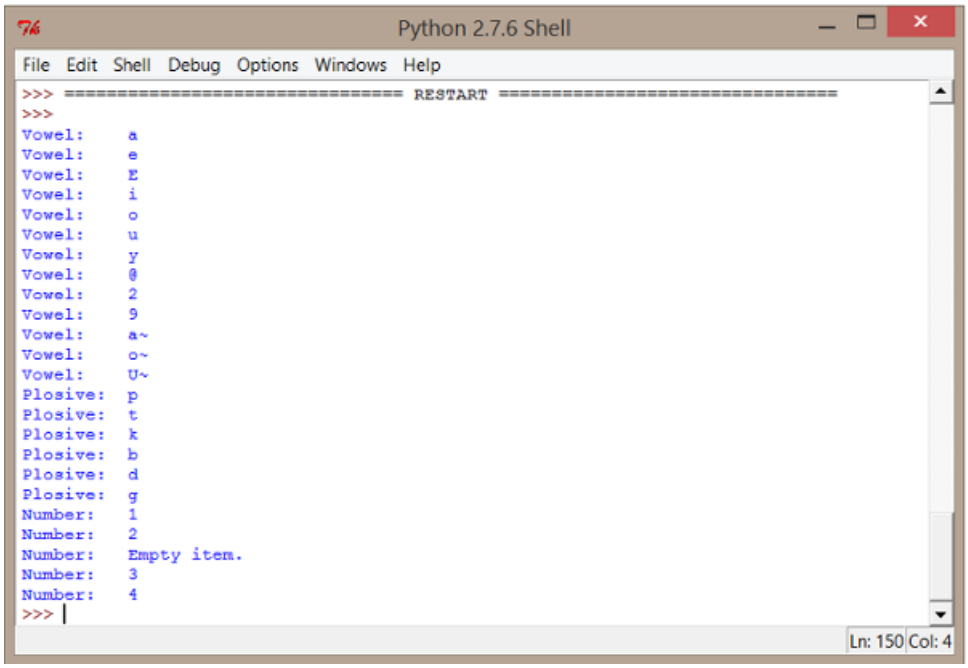


Figure 6.5: Expected output of the 3rd script

6.3.4 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 this lines of code in a new script and try it (solution: ex04_reading_simple.py)

```
fp = open("C:\Users\Me\Desktop\pythonscripts\phonemes.csv", 'r')
for line in fp:
    # do something with the line stored in variable l
    print(line.strip())
f.close()
```

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

```
def read_file(filename):
    """ Get the content of file.

    :param filename: (string) Name of the file to read, including path.
    :returns: List of lines

    """
    with codecs.open(filename, 'r', encoding="utf8") as fp:
        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 )

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)
```

6.3.5 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

6.3.6 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 The API of SPPAS to manage data

In this section, we are going to create Python scripts with the SPPAS Application Programming Interface *annotationdata* and then run them.

6.4.1 Overview

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

First of all, it is important to understand the data structure included into the API to be able to use it efficiently. Details can be found in the following publication:

> *Brigitte Bigi, Tatsuya Watanabe, Laurent Prévot* (2014). **Representing Multimodal Linguistics Annotated data**, Proceedings of the 9th edition of the Language Resources and Evaluation Conference, 26-31 May 2014, Reykjavik, Iceland.

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 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 *annotationdata* API was designed to be able to manipulate all data in the same way, regardless of the file type. The API 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 `Transcription` object. A `Transcription` has a name, and a list of `Tier` objects. Two tiers can't share the same name. The list of tiers can be empty. 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 constituency.

A `Tier` has a name, and a list of `Annotation` objects. It also contains a set of meta-data and it can be associated to a controlled vocabulary.

An annotation is made of 2 objects:

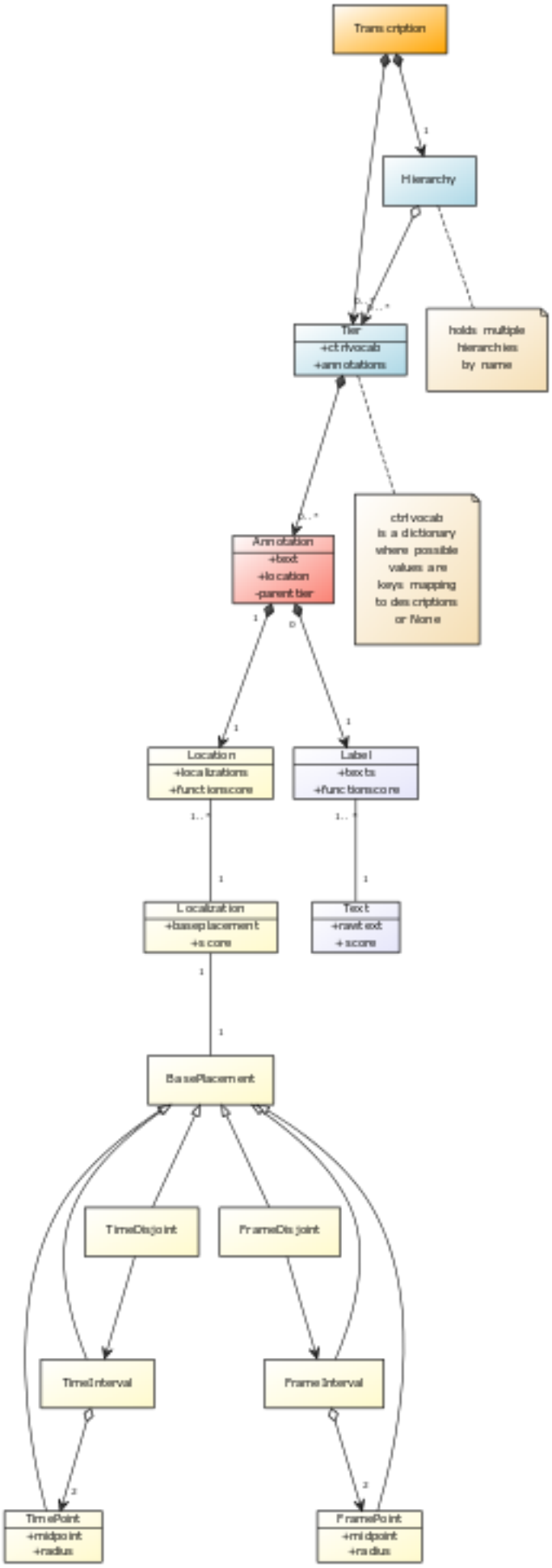
- a `Location` object,
- a `Label` object.

A `Label` object is representing the "content" of the annotation. It is a list of `Text` associated to a score.

A `Location` is representing where this annotation occurs in the media. Then, a `Location` is made of a list of `Localization` which includes the `BasePlacement` associated with a score. A `BasePlacement` object is one of:

- a `TimePoint` object; or
- a `TimeInterval` object, which is made of 2 `TimePoint` objects; or
- a `TimeDisjoint` object which is a list of `TimeInterval`; or
- a `FramePoint` object; or
- a `FrameInterval` object; or
- a `FrameDisjoint` object.

The whole API documentation is available at the following URL: http://www.sppas.org/manual/module-tree.html


Transcription
Hierarchy
Tier
+ctrlvocab
+annotations
holds multiple hierarchies by name
Annotation
+text
+location
-parent tier
ctrl vocab is a dictionary where possible values are keys mapping to descriptions or None
Location
+localizations
+functionscore
Label
+texts
+functionscore
Localization
+baseplacement
+score
Text
+rawtext
+score
BasePlacement
TimeDisjoint
FrameDisjoint
TimeInterval
FrameInterval
TimePoint
+midpoint
+radius
FramePoint
+midpoint
+radius

Label representation

Each annotation holds at least one label, mainly represented in the form of a string, freely written by the annotator or selected from a list of categories, depending on the annotation tool. The *annotationdata* API allows to assign a score to each label, and allows multiple labels for one annotation.

Location representation

In the *annotationdata* API, a `TimePoint` 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.

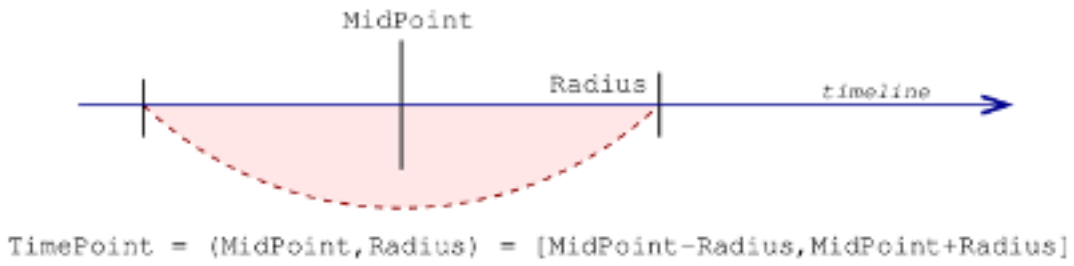


Figure 6.7: Representation of a TimePoint

Example

The screenshot below shows an example of multimodal annotated data, imported from 3 different annotation tools. Each `TimePoint` 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.

6.5 Creating scripts with annotationdata API

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!

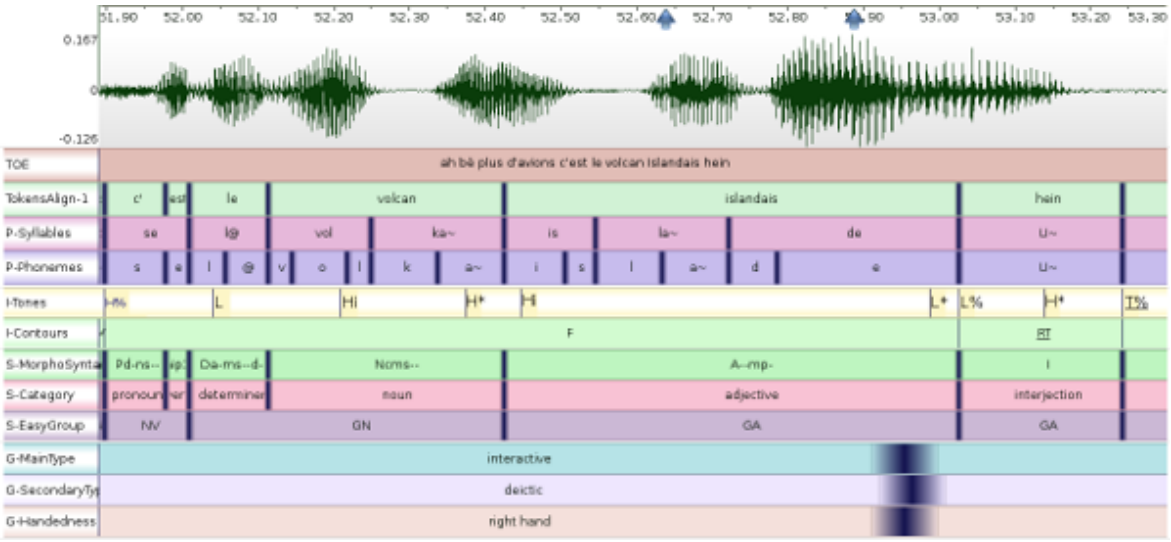


Figure 6.8: Example of multimodal data

6.5.2 Read/Write annotated files

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

```
trs = trsio.read(filename_in)
trsaio.write(filename_out, trs)
```

Only these two lines of code are required to convert a file from one format to another one! The appropriate reader/writer for the format is given by the extension of the name of the file.

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

Currently, accepted input file extensions are:

- xra
- csv, txt
- TextGrid, PitchTier, IntensityTier
- eaf
- trs
- mrk
- sub, srt
- hz
- antx
- tdf

Possible output file extensions are:

- xra
- csv, txt, lab, ctm, stm

- TextGrid, PitchTier
- eaf
- mrk
- sub, srt
- antx

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

6.5.3 Manipulating a Transcription object

The most useful functions used to manage a Transcription object are:

- `Append(tier),Pop()`
- `Add(tier, index),Remove(index)`
- `Find(name, case_sensitive=True)`

`Append()` is used to add a tier at the end of the list of tiers of the Transcription object; and `Pop()` is used to remove the last tier of such list.

`Add()` and `Remove()` do the same, except that it does not put/delete the tier at the end of the list but at the given index.

`Find()` is useful to get a tier from its name.

```
trs = trsaio.read('Filename-palign.TextGrid)
for tier in trs:
    # do something with the tier:
    print(tier.GetName())
phonemes_tier = trs.Find("PhonAlign")
```

Transcription object has an iterator to get access to tiers.

```
trs = trsaio.read('Filename-palign.TextGrid)
phonemes_tier = trs[0]
```

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 Tier object

A tier is made of a name, a list of annotations and meta-data. To get the name of a tier, or to fix a new name, the easier way is to use `tier.GetName()`.

The following block of code allow to get a tier and change its name. It should be tested into a script…

```
trs = trsaio.read(filename)
tier = trs[0]
print(tier.GetName())
tier.SetName("toto")
print(tier.GetName())
print(trs[0].GetName())
```

The most useful functions to manage a Tier object are:

- `Append(annotation)`, `Pop()`
- `Add(annotation)`, `Remove(begin, end, overlaps=False)`
- `IsDisjoint()`, `IsInterval()`, `IsPoint()`
- `Find(begin, end, overlaps=True)`
- `Near(time, direction)`
- `SetRadius(radius)`

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

Goodies:

The file `ex12_tiers_info_wx.py` proposes a GUI to print information of one file, or all files of a directory, and to ask the file/directory name with a dialogue frame, instead of fixing it in the script. This script can be executed simply by double-clicking on it in the File Explorer of your system. Many functions of this script can be cut/pasted in any other script.

6.5.5 Main information on Annotation/Location/Label objects

The most useful methods used to manage an `Annotation` object are:

- `IsSilence()`, `IsLabel()`
- `IsPoint()`, `IsInterval()`, `IsDisjoint()`
- `GetBegin()`, `SetBegin(time)`, only if time is a TimeInterval
- `GetEnd()`, `SetEnd(time)`, only if time is a TimeInterval
- `GetPoint()`, `SetPoint(time)`, only if time is a TimePoint

The following example shows how to get/set a new label, and to set a new time value to an annotation:

```
if ann.GetLabel().IsEmpty():
    ann.GetLabel().SetValue("dummy")
if ann.GetLocation().IsPoint():
    p = ann.GetLocation().GetPoint()
    p.SetMidPoint(0.234)
    p.SetRadius(0.02)
if ann.GetLocation().IsInterval():
    ann.GetLocation().GetBegin().SetMidPoint(0.123)
    ann.GetLocation().GetEnd().SetMidPoint(0.234)
```

If something forbidden is attempted, the object will raise an Exception. This means that the program will stop except if the script "raises" the exception.

6.5.6 Exercises

Exercise 1: Write a script to print information about annotations of a tier (solution: ex13_tiers_info.py)

Exercise 2: Write a script to estimates the frequency of a specific annotation label in a file/corpus (solution: ex14_freq.py)

6.5.7 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 `Tier()` class. The idea is that each `Tier()` 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 `Transcription()` 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 depending on the annotation type.

Creating a boolean function

In the following, let `Bool` and `Rel` two predicates, a tier `T`, and a filter `f`.

Thus, the following matching predicates are proposed to select annotations (intervals or points) depending on their label. Notice that `P` represents the text pattern to find:

- exact match: `pr = Bool(exact=P)`, means that a label is valid if it strictly corresponds to the expected pattern;
- contains: `pr = Bool(contains=P)`, means that a label is valid if it contains the expected pattern;
- starts with, `pr = Bool(startswith=P)`, means that a label is valid if it starts with the expected pattern;
- ends with, `pr = Bool(endswith=P)`, means that a label is valid if it ends with the expected pattern.

These predicates are then used while creating a filter on labels. All these matches can be reversed, to represent does not exactly match, does not contain, does not start with or does not end with.

The next examples illustrate how to work with such filters and patterns. 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")
ft = Filter(tier)
f1 = LabelFilter(Bool(exact='a'), ft)
f2 = LabelFilter(~Bool(iexact='a'), ft)
```

For complex search, a selection based on regular expressions is available by using `pr = Bool(regexp=R)`.

A multiple pattern selection can be expressed with the operators `|` to represent the logical "or" and the operator `&` to represent the logical "and".

With this notation in hands, it is possible to formulate queries as, for example: *Extract words starting by "ch" or "sh"*, like:

```
pr = Bool(startswith="ch") | Bool(startswith="sh")
```

Filters on duration can also be created on annotations if Time instance is of type TimeInterval. In the following, `v` represents the value to be compared with:

- lower: `pr = Bool(duration_lt=v)`, means that an annotation of `T` is valid if its duration is lower than `v`;
- lower or equal: `pr = Bool(duration_le=v)`;
- greater: `pr = Bool(duration_gt=v)`;
- greater or equal: `pr = Bool(duration_ge=v)`;
- equal: `pr = Bool(duration_e=v)`;

Search can also starts and ends at specific time values in a tier by creating filters with `begin_ge` and `end_le`.

Finally, the user must apply the filter to get filtered data from the filter.

```
# creating a complex boolean function
predicate = (Bool(icontains="a") | Bool(icontains="e")) & Bool(duration_ge=0.08)

# to create a filter:
ft = Filter(tier)
flab = LabelFilter(predicate, ft)

# to get filtered data from the filter:
tier = flab.Filter()
tier.SetName('Filtered with a-e-0.8')
```

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.

We implemented the 13 Allen interval relations: before, after, meets, met by, overlaps, overlapped by, starts, started by, finishes, finished by, contains, during and equals. Actually, we implemented the 25 relations proposed 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).

MISSING:List of Allen interval relations./etc/screenshots/allen.png

Below is an example of implementing the request: *Which syllables stretch across 2 words?*

```
# Get tiers from a Transcription object
tiersyll = trs.Find("Syllables")
tiertoks = trs.Find("TokensAlign")

# Create filters
fsyll = Filter(tiersyll)
ftoks = Filter(tiertoks)

# Create the filter with the relation function (link both filters)
predicate = Rel("overlaps") | Rel("overlappedby")
f = RelationFilter(relation, fsyll, ftoks)

# to get filtered data from the filter:
tier = f.Filter()
tier.SetName('Syllables across Tokens')
```

6.5.8 Exercises

Exercise 1: Create a script to filter annotated data on their label (solution: ex15_annotation_label_filter.py).

Exercise 2: Idem with a filter on duration or time. (solution: ex16_annotation_time_filter.py).

Exercise 3: Create a script to get tokens followed by a silence. (solution: ex17_annotations_relation_filter1.py).

Exercise 4: Create a script to get tokens preceded by OR followed by a silence. (solution: ex17_annotations_relation_filter2.py).

Exercise 5: Create a script to get tokens preceded by AND followed by a silence. (solution: ex17_annotations_relation_filter3.py).

6.6 More with SPPAS API

In addition to *annotationdata*, SPPAS contains several other API that could be relevant for users to simplify their lives!!! 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.

7

References

7.1 References about SPPAS

7.1.1 How to cite SPPAS?

By using SPPAS, you agree to cite any of the references in this chapter. For a general citation of SPPAS, simply use the main reference below. A PDF version of this publication is available in the folder `documentation` of the SPPAS 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, 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 ans Resources for the Analysis of Speech Prosody, Aix-en-Provence, France, pp. 62-65.

7.1.3 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).

7.1.4 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.

7.1.5 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.

7.1.6 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.

7.1.7 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), pages 2648-2652. ISBN: 978-2-9517408-8-4.

7.1.8 About analyses tools

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

7.1.9 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.

7.1.10 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.

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, Jaèn (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**, Proceedins of Tools ans 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**, Proceedins of Tools ans 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.

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 ans 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, morphosyntaxic, syntaxic, 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.

8

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 then 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.
- IPUs 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 main changes:

- 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 i 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

Annotations:

- IPUs 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.

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 IPUs** 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)

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)

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

Tools/scripts:

- tierfilter.py
- tiercombine.py

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

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)

Requests/Filter:

- "Starts search at…" and "Ends search at…"
- remove negative search (because of a bug…). Replaced by a "reverse" option.

Development:

- re-organization of lib, except for the wxGUI library.
- Cancel the sppas.lock file creation when SPPAS is running.

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 syllabes 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)

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, doc-strings, 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 childhood 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
- IPUs 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
  - sclite.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 christmast 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 wxqui/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.

9

Appendix

9.1 List of error messages

Since version 1.9.0, SPPAS introduced a system for internationalization of the messages. In the same time, a quality and a number was assigned progressively to each of them in the packages. The following indicates the list of error messages that can occur while using SPPAS.

From "annotations" package

:ERROR 1010: Unknown option with key {key}.

:ERROR 1020: Empty input tier {name}.

:ERROR 1030: Missing input tier. Please read the documentation.

:ERROR 1040: Bad input tier type. Expected time-aligned intervals.

:ERROR 1050: Inconsistency between the number of intervals of the input tiers. Got: {:d} and {:d}.

From "utils" package

:ERROR 1210: The directory {dirname} does not exist.

:ERROR 1220: The directory {dirname} does not contain relevant data.

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}.

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}].

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}.

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}.

: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}).

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}.

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}.

: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. Secondarily, 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 worldwide, 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.