SPPAS: a tool for the phonetic segmentation of speech

Brigitte Bigi

Laboratoire Parole et Langage
CNRS & Aix-Marseille Université,
5, avenue Pasteur, BP80975
13604 Aix-en-Provence France
brigitte.bigi@lpl-aix.fr

Abstract

SPPAS is a tool to produce automatic annotations which include utterance, word, syllabic and phonemic segmentations from a recorded speech sound and its transcription. SPPAS is distributed under the terms of the GNU Public License. It was successfully applied during the Evalita 2011 campaign, on Italian map-task dialogues. It can also deal with French, English and Chinese and there is an easy way to add other languages. The paper describes the development of resources and free tools, consisting of acoustic models, phonetic dictionaries, and libraries and programs to deal with these data. All of them are publicly available.

Keywords: phonetic, annotation, segmentation

1. Introduction

The last ten years or so have witnessed an explosion in the quantity of linguistic data which have become available as evidence for the nature of linguistic representations of speech. Whereas it was common some years ago to formulate phonetic models on the basis of rather limited data, today it is becoming more and more expected for linguists to take into account large quantities of empirical data, often including several hours of recorded speech. Up until a few years ago it was difficult, if not impossible, to manipulate such large quantities of data outside specialized laboratories using expensive main-frame computers. Software for the analysis of acoustic data was often difficult to install and use, nearly always platform specific and, above all, very expensive. Today, the situation has radically changed and the availability of cheaper and cheaper data storage has made it possible to envisage data analysis involving several hundreds gigabytes data. In the same period, a number of software tools for the analysis of audio and/or video recordings of speech have become available such as Anvil (Kipp, 2001), Elan (Sloetjes et al., 2010), Praat (Boersma and Weenink, 2009), Transcriber (TranscriberAG, 2011) and WaveSurfer (WaveSurfer, 2012), to name just some of the most popular tools, all of which are both free and multi-platform. For an extensive list of speech analysis software see (Llisterri, 2011). The biggest obstacle linguists are faced with today is not the storage of data, nor its analysis, but its annotation.

The analysis of the phonetic entities of speech nearly always requires the alignment of the speech recording with a phonetic transcription of the speech. This task is extremely labour-intensive - it may require several hours for even an experienced phonetician to transcribe and align a single minute of speech manually. It is consequently obvious that transcribing and aligning several hours of speech by hand is not generally something which can be envisaged.

A number of tool boxes are currently available which can be used to automate the task, including the HTK Toolkit (Young and Young, 1994), Sphinx (Carnegie Mellon University, 2011), or Julius (Nagoya Institute of Technology, 2010). For most of these, our experience is that the tools require a level of expertise in computer science which is beyond the reach of most linguists without the help of an engineer. Some wrapper tools propose to simplify the use of HTK, as for example P2FA (Yuan and Liberman, 2008) or EasyAlign (Goldman, 2011). The EasyAlign Praat plugin is currently the most “linguist-friendly” but it currently runs only on Windows. Moreover, due to the HTK license limitations, both tools assume that HTK is installed on the user’s computer.

SPPAS is a tool to produce automatic annotations which include utterance, word, syllabic and phonemic segmentations from a recorded speech sound and its transcription. The whole procedure is a succession of 4 automatic steps and resulting alignments are a set of TextGrid files. TextGrid is the native file format of the Praat software which became one of the most common tool for phoneticians (Boersma and Weenink, 2009). *SPPAS* is implemented with python and was tested under Linux, Mac-OSX and Windows®. *SPPAS* is currently designed for French, English, Italian and Chinese and there is an easy way to add other languages. Recently, it was successfully applied to the forced alignment task during the Evalita 2011 campaign, on Italian map-task dialogues. Evalita is an initiative devoted to the evaluation of Natural Language Processing and Speech tools for Italian[1]. Systems were required to align audio sequences of spoken dialogues to the provided relative transcriptions.

This tool and all resources are distributed under the terms of the GPL license at the URL:

http://www.lpl-aix.fr/∼bigi/sppas/

SPPAS is also designed to be used directly by linguists. Current developments consist in integrating Momel (Hirst and Espesser, 1993) and Intsint (Hirst and Di Cristo, 1998). Section 2. of this paper presents an overview of what SPPAS (version 1.4) can do. Section 3. describes resources included in SPPAS. Finally, Section 4. gives details about the SPPAS architecture.

[1] http://www.evalita.it/

2. SPPAS overview

2.1. Main description

SPPAS can be executed using the GUI - Graphical User Interface, or by using tools in a console-mode. Figures 6 and 7 show the GUI of version 1.3 and version 1.4 respectively. To execute SPPAS, the user just need to double-click on the *SPPAS* main program. A wav file or a directory containing a set of wav files and the corresponding transcriptions have to be selected. Then, the user selects the appropriate language and modules to execute. The Preferences button allows to configure these modules (to fix options).

Figure 1 summarizes the workflow, with all possible inputs/outputs. Inputs are represented in yellow boxes and black arrows. Outputs are represented with the green boxes and the green arrows. *SPPAS* can be used in a completely automatic mode. It can also be used semi-automatically, by respecting file name conventions and transcription conventions.

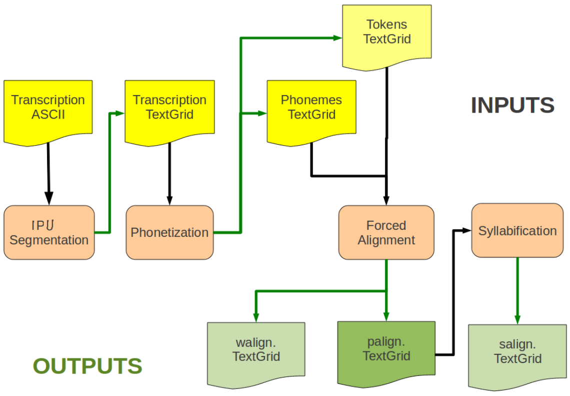


Figure 1: SPPAS workflow

2.2. History of the versions

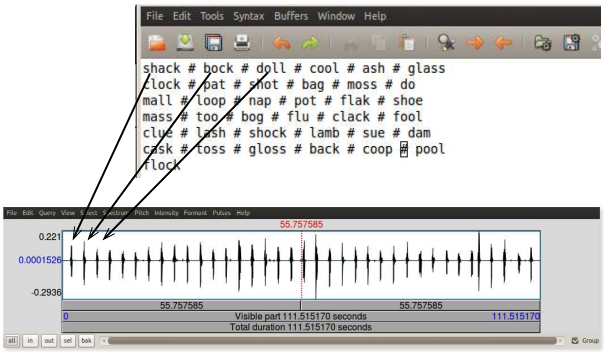
The first version of *SPPAS* was released on March 9, 2011. This version was made only of *tcsh* and gawk scripts and it could be used only in a console mode. In this first version, no resources were included. Some resources were added in version 1.1 (in June, 2011). The next release was distributed on July 23, 2011. This version included resources to deal with English, French, Italian (read speech, monophones) and Chinese. MacOS support was added in version 1.3, and the first GUI was included in this version. Italian acoustic models and the dictionary was changed and the French acoustic models was improved.

Version 1.4 will to be released on May, 2012. This version will be entirely implemented with python and far easier to install than previous versions. The present paper describes this version.

2.3. Inter-Pausal Unit Segmentation

Inter-pausal unit segmentation is an open research problem. It consists in aligning macro-units of a document with the corresponding sound, as illustrated in Figure 2. The implementation of SPPAS - version 1.4, includes an algorithm for this step which we hope to improve in future versions of the software.

In *SPPAS*, silent pauses can be indicated in the transcription either by the symbol '#' or by a newline, or both. A recorded speech file with the .wav extension must correspond to each .txt file. The correspondences are established by means of the file names.


File Edit Tools Syntax Buffers Window Help

```
shack # bock # doll # cool # ash # glass
clock # pat # shot # bag # moss # do
mall # loop # nap # pot # flak # shoe
mass # too # bog # flu # clack # fool
clue # lash # shock # lamb # sue # dam
cask # toss # gloss # back # coop # pool
flock
```

File Edit Query View Select Spectrum Pitch Intensity Formant Pulses Help
55.757585
0.221
0.0001526
-0.2936
55.757585
55.757585
0
Visible part 111.515170 seconds
111.515170
Total duration 111.515170 seconds
all
in
out
sel
bak
Group

Figure 2: Example of IPU-segmentation

The algorithm currently implemented identifies silent pauses in the signal and attempts to align them with the inter-pausal units proposed in the transcription, under the assumption that each such unit is separated by a silent pause. For a given minimum duration for pauses and for inter-pausal units a dichotomic search adjusts the silence threshold (in dB) and identifies the number of units thus defined. If the number of units found is greater or smaller than the desired number, the search is renewed adjusting the minimum duration of the silences and units accordingly. The search halts when the three parameters are correctly adjusted: minimal duration of pauses, minimal duration of units and silence threshold.

This ipu-segmentation was used to align read speech in two projects. The first one consisted in a set of recordings of 40 English words. SPPAS succeeded to align all files. The second one consisted of a set of recordings of paragraphs made of 3 to 6 sentences. These are part of the AixOx corpus (Herment et al., 2012). In this case, SPPAS failed to align about 10% of the files and 40% of the sentence boundaries had to be manually corrected. Errors were only due to the fact that silences were not at the end of sentences as it was supposed to be in the script speakers had to read. To perform this ipu-segmentation, and despite these errors, the use of SPPAS saved time.

2.4. Phonetization

Phonetization is the process of representing sounds with phonetic signs. There are two general ways to construct a phonetization process: dictionary based solutions which consist in storing a maximum of phonological knowledge in a lexicon and rule based systems with rules based on inference approaches or proposed by expert linguists. In both cases, phonetization is based on a manual transcription.

Clearly, there are different ways to pronounce the same utterance. Different speakers have different accents and tend to speak at different rates. When a speech corpus is transcribed into a written text, the transcriber is immediately confronted with the following question: how to reflect the orality of the corpus? Conventions are then designed to provide rules for writing speech corpora. These conventions establish phenomena to annotate and also how to annotate them.

A system based on a dictionary solution consists in storing a maximum of phonological knowledge in a lexicon. In this sense, this approach is language-independent unlike rule-based systems. The SPPAS module for the phonetization of the orthographic transcription produces a phonetic transcription based on a phonetic dictionary. The phonetization is the equivalent of a sequence of dictionary look-ups. It is assumed that all words of the speech transcription are mentioned in the pronunciation dictionary. Otherwise, SPPAS implements a language-independent algorithm to phonetize unknown words. At this stage, it consists in exploring the unknown word from left to right and to find the longest strings in the dictionary. Since this algorithm uses the dictionary, the quality of such a phonetization will depend on this resource.

Actually, some words can correspond to several entries in the dictionary with various pronunciations. Unlike rule-based systems, in dictionary-based solutions the pronunciation is not supposed to be “standard”. Phonetic variants are proposed for the aligner to choose the phoneme string. The hypothesis is that the answer to the phonetization question is in the signal. For example, the French sentence *je suis* can be pronounced:

- /ʒsɥi/ is the standard pronunciation,
- /ʒsɥiz/ is the standard pronunciation plus a liaison,
- /ʒəsɥi/ is the South of France pronunciation,
- /ʒəsɥiz/ is the South of France pronunciation plus a liaison,
- /ʃɥi/ is a frequent specific realization.

The corresponding dictionary entries for both words are:

je [je] ʒ

je(2) [je] ʒə

je(3) [je] ʃ

suis [suis] sɥi

suis(2) [suis] sɥiz

suis(3) [suis] sui

suis(4) [suis] ɥi

suis(5) [suis] ɥiz

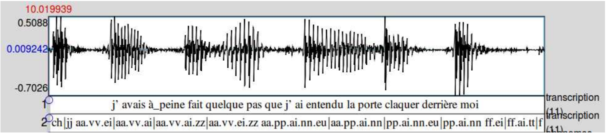
SPPAS can take as input a tokenized standard orthographic transcription and some enrichments only if the acoustic model includes them. For example, the French transcriptions can contain laugh (represented by the symbol ’@’ in the transcription).

An example of *SPPAS* phonetization output is presented in Figure 3 on a French sentence. By using *SPPAS* in a semi-automatic way, this phonetization can be manually modified. The *SPPAS* conventions are:

- spaces separate tokens,
- dots separate phonemes,
- pipes separate phonetic variants.

Thus, the phrase *“je suis”* is phonetized as:

ʒ|ʒ.ə|ʃ s.ɥ.i|s.ɥ.i.z|s.u.i|ɥ.i|ɥ.i.z



j’ avais à_peine fait quelque pas que j’ ai entendu la porte claquer derrière moi

ch|jj aa.vv.ei|aa.vv.ai|aa.vv.ai.zz|aa.vv.ei.zz aa.pp.ai.nn.eu|aa.pp.ai.nn|pp.ai.nn.eu|pp.ai.nn ff.ei|ff.ai.tt|f

Figure 3: Example of Phonetization (French)

2.5. Alignment

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. The goal is to generate an alignment between the speech signal and its phonetic representation.

SPPAS is based on the Julius Speech Recognition Engine (SRE) (Nagoya Institute of Technology, 2010) for three reasons: 1/ it is easy to install which is important for users; 2/ it is also easy to use then easy to integrate in SPPAS; and 3/ its performances correspond to the state-of-the-art of HMM-based systems and are quite good. Julius was designed for dictation applications, however the Julius distribution only includes Japanese Acoustic Models. But since it uses Acoustic Models trained using the HTK toolkit (Young and Young, 1994), it can also use Acoustic Models trained for other languages. Initially, Julian was a special version of Julius that performed grammar based speech recognition. The release 4 merged Julius and Julian.

To perform alignment, a finite state grammar that describes sentence patterns to be recognized and an acoustic model are needed. A grammar essentially defines constraints on what the SRE can expect as input. It is a list of words that the SRE listens to. Each word has a set of associated list of phonemes, extracted from the dictionary. When given a speech input, Julius searches for the most likely word sequence under constraint of the given grammar. For example, the sentence “we are reading” will produce the following list of words with associated pronunciations (extracted from the CMU pronunciation dictionary):

```
0 [w_0] w iy
0 [w_0] w ih
1 [w_1] aa r
1 [w_1] er
2 [w_2] r eh dx ix ng
2 [w_2] r iy dx ix ng
```

The Julius corresponding grammar file is defined as:

```
0 2 1 0 1
1 1 2 0 0
2 0 3 0 0
3 -1 -1 1 0
```

The grammar indicates that each word of the dictionary has to be aligned, from word 0 to word 1, from word 1 to word 2, and so on. The Julius alignment task is a 2-step process: the first step chooses the phonetization and the second step performs the segmentation.

Speech Alignment also 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. *SPPAS* is based on the use of HTK-ASCII acoustic models. Then, acoustic models were trained with HTK by taking the training corpus of speech, previously segmented in to utterances and phonetized. HTK (Hidden Markov Toolkit) is a portable toolkit for building and manipulating the statistical models used to represent sound in Speech Recognition. These are called Hidden Markov models (HMMs). HTK's licence requires registering before it can be downloaded. The software is open source but there are limitations on the distribution of the HTK Toolkit itself. For example, an HTK tool can not be embedded in an other tool. However, there is no limitation on the distribution of the models created with the toolkit. This model can be distributed under GPL license.

Figure 4 illustrates a SPPAS alignment output on a French sentence. The alignment of words is deduced from their alignment into phonemes.

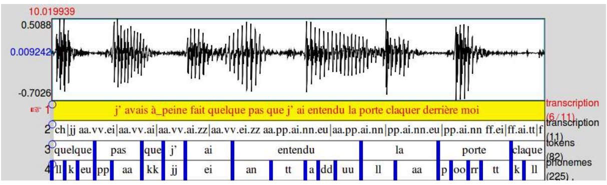

10.019939
0.5088
0.009242
-0.7026
1
j' avais à_peine fait quelque pas que j' ai entendu la porte claquer derrière moi
transcription (6/11)
2
ch|jj aa.vv.ei|aa.vv.ai|aa.vv.ai.zz|aa.vv.ei.zz aa.pp.ai.nn.eu|aa.pp.ai.nn|pp.ai.nn.eu|pp.ai.nn ff.ei|ff.ai.tt|f
transcription (11)
3
quelque pas que j' ai entendu la porte claque
tokens (82)
4
ll k eu pp aa kk jj ei an tt a dd uu ll aa p oo rr tt k ll
phonemes (225)

Figure 4: Example of alignment (French)

2.6. Syllabification

The syllabification of phonemes is performed with a rule-based system previously described for French in (Bigi et al., 2010). This RBS phoneme-to-syllable segmentation system is based on 2 main principles:

1. a syllable contains a vowel, and only one.
2. a pause is a syllable boundary.

These two principles focus the problem of the task of finding a syllabic boundary between two vowels. As in state-of-the-art systems, phonemes were grouped into classes and rules established to deal with these classes. This system was applied for the labelling of CID (Bertrand et al., 2008). Evaluations were carried out by using a subset of 7 minutes. Two experts labelled this corpus manually. The syllable agreement rate was 97.8% between experts. This agreement was 95.8% and 94.9% between automatic system and each expert.

A new set of rules was developed to deal with Italian. At present there is, to our knowledge, no generally available corpus manually labelled to evaluate such a syllabification. We will of course be pleased to perform this evaluation as soon as we can obtain one.

3. SPPAS Resources

3.1. Phonetic resources

An important step is building pronunciation dictionaries, where each word in the vocabulary is expanded into its constituent phones.

The dictionary file format is the same as that used in the HTK toolkit (Young and Young, 1994) and in the VoxForge project (VoxForge, 2006 2011), as for example:

```
A [A] a
A(2) [A] wa
A(3) [A] ja
ACQUA [ACQUA] a k wa
ACQUA(2) [ACQUA] k wa
```

The dictionary contains a set of possible pronunciations for words, including accents as *perchè* pronounced as /b e r k e/, and reduction phenomena as /p e k/ or /k wa/ for the word *acqua*.

The Italian dictionary was downloaded from the Festival synthetizer tool (The Centre for Speech Technology Research, 2011). This dictionary was enriched by word pronunciations observed in the Evalita train corpus. A large set of both phonetizations was manually corrected. Finally, the dictionary contains about 390k words and 5k variants.

The French dictionary was constructed by merging several free dictionaries. Some word pronunciations were also added using the LIA_Phon tool. It contains about 350k words and 300k variants.

The English dictionary *SPPAS* deals with was downloaded from the VoxForge project (VoxForge, 2006 2011).

The Chinese dictionary was hand-made. It now contains about 350 entries and is still in progress.

3.2. Acoustic Model (AcM) Training

SPPAS is based on a common practice and uses context-independent HMMs for speech segmentation. *SPPAS* can use various types of acoustic models. This section describes how we created models which are included in *SPPAS*. The Italian, French and Chinese models were trained following these steps. An English model can be downloaded from the VoxForge web site.

French AcM was trained by using 7h30 of conversational speech extracted from CID (Bertrand et al., 2008), and 30 minutes of read speech extracted from AixOx corpus (Herment et al., 2012). Italian AcM was trained from 3h30 of map-tasks dialogues distributed during the Evalita 2011 campaign. Chinese AcM was trained from 50 min read speech, from Chinese Multext corpus.

The phoneme statistical representation is based on a 5-state model with a left-to-right topology with self-loops and no

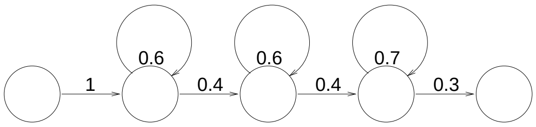


Figure 5: 5-states HMM with initial probabilities

transitions which skip over states, as represented in Figure 5 with its initial probabilities.

Ideally, the phones would have unique articulatory and acoustic correlates. But acoustic properties of a given phone can depend on the phonetic environment. These co-articulation phenomena motivated the adoption of context-dependent models such as triphones, except for Chinese.

Our training procedure was adapted from the VoxForge tutorial. Typically, the HMM states are modelled by Gaussian mixture densities whose parameters are estimated using an expectation maximization (EM) procedure. The outcome of this training procedure is dependent on the availability of accurately annotated data and on good initialization. As more speech audio data is collected, better Acoustic Models can be created. Acoustic models were trained from 16 bits, 16000 hz wav files. The Mel-frequency cepstrum coefficients (MFCC) along with their first and second derivatives were extracted from the speech in the standard way.

The acoustic model training procedure is based on 3 main steps. Step 1 is the data preparation. It establishes the list of phonemes, plus silence and short pauses. It converts the input data (phonetization of the corpus) into the HTK-specific data format. It codes the (audio) data: this step is called "parameterizing the raw speech waveforms into sequences of feature vectors" (from wav format to MFCC). Step 2 is the monophones generation. It creates a Flat Start Monophones model by defining a prototype model and copying this model for each phoneme. Then, this flat model is re-estimated using the MFCC files to create a new model. Then, it fixes the “sp” model from the “sil” model by extracting only 3 states of the initial 5-states model. Then, this model is re-estimated using the MFCC files and the phonetization. Step 3 creates tied-state triphones from monophones and from some language specificities defined by the way of a configuration file. This file summarizes phonemic informations as for example the list of vowels, liquids, fricatives, nasals or stops. This resource was created manually for Italian, French and Chinese and are available on-demand.

4. SPPAS Architecture

4.1. SPPAS 1.4 packages

SPPAS is implemented using the programming language *Python*. This choice was motivated by many reasons. First of all, “All Python releases are Open Source (see http://www.opensource.org/ for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible” (http://docs.python.org/license.html). For many operating systems, Python is a standard component. Thanks to being scripting language with module architecture, syntax simplicity and rich text processing tools, Python is object-oriented programming.

SPPAS version 1.4 package content is detailed below:

README		Several general information
CHANGES		Versions tracking
COPYRIGHT		GPL license
dict		Pronunciation dictionaries
	FR.dict	French
	IT.dict	Italian
	EN.dict	English
	ZH.dict	Simplified Chinese
models		Acoustic models
	models-FR	French: hmmdefs and tiedlist
	models-IT	Italian: hmmdefs and tiedlist
	models-EN	English: hmmdefs and tiedlist
	models-ZH	Simplified Chinese: hmmdefs
syll		Syllabification configurations
	syllConfig-FR.txt	French
	syllConfig-IT.txt	Italian
samples		A set of samples
...		(wav and transcriptions)
	samples-FR	French set of examples
	samples-IT	Italian set of examples
	samples-EN	English set of examples
	samples-ZH	Simplified Chinese
lib		SPPAS library
	param.py	sppasParam Class
	log.py	sppasLog Class
	wav	
	trs	
	phon	
	align	
	momel	
	gui	
	term	
tools		SPPAS tools
	wavsplit.py	
	phonetize.py	
	alignment.py	
	LPL-Syllabeur-2.2.jar	
	momel.py	
...		
sppas.command		Bash to execute SPPAS (Unix-based systems)
sppas.py		Main SPPAS program

SPPAS can deal with a new language L by simply adding the language resources to the appropriate directories:

- the dictionary to: SPPAS/dict/L.dict
- the acoustic model to: SPPAS/models/models-L
- the syllabification rules to: SPPAS/syll/syllConfig-L.txt

By using the GUI, *SPPAS* dynamically creates the list of available languages by exploring appropriate directories. Steps using language-dependent resources (phonetization, alignment and syllabification) are activated or disabled depending on the directory contents.

4.2. SPPAS 1.4 modules

SPPAS library is placed in the "lib" directory. Shared modules are placed directly in this directory. It is the case for example of the Class "sppasLog" that handles the log file created by SPPAS during each run. Sub-directories are used to separate SPPAS modules. For each one of these modules, a set of classes is implemented:

wav This module contains a set of classes dealing with wav files: the main class "Wave", the class "WavePitch", the class "WaveSil".

trs This module contains a set of classes dealing with a "Transcription" represented as a set of "Tier" instances. A tier is a set of "Annotation" instances. Annotations are represented by a label and 1 or 2 time values, depending on the annotation type: interval or point. A set of Transcription Input/Output classes are also available and are still in-progress: current version includes a TextGrid IO class and an Ascii IO class (txt or csv).

phon This module contains a set of classes dealing with the phonetization problem. The class "DictPhon" is the main class to perform phonetization; "PhonUnk" is related to the phonetization of unknown words; and "PhonTrs" uses DictPhon on a whole tier.

align This module contains a set of classes dealing with the alignment problem. It is a 3-steps process: 1/ the transcription and related wav files are split into units; 2/ each unit is aligned; then 3/ unit alignments are merged in a transcription tier and saved. Step 2 is a little bit more complicated than it can look at first. We encountered 2 difficulties using Julius. Firstly, a unit is not aligned if a triphone is missing in the tiedlist. We implemented a "TiedList" class that adds the unobserved triphone into the tiedlist. Secondly, we observed that the alignment failed for about 3-5% of units. In these cases, Julius performs its 1st step properly (it chooses the phonetization, depending on the grammar) but its 2nd step (the segmentation) fails. To deal with these errors (and to produce a result instead of nothing), we implemented a function that uses the same duration for each phone of the unit.

momel This module contains the SPPAS implementation of the Momel (Hirst and Espesser, 1993) and Intsint (Hirst and Di Cristo, 1998) tools. Momel (Modelling melody) is an algorithm for the analysis and synthesis of intonation patterns. INTSINT is an acronym for INternational Transcription System for INTonation. INTSINT codes the intonation of an utterance by means of an alphabet of 8 discrete symbols constituting a surface phonological representation of the intonation: T (Top), H (Higher), U (Upstepped), S (Same), M (mid), D (Downstepped), L (Lower), B (Bottom).

gui All classes related to the graphical user interface.

term All classes related to the terminal user interface.

These modules constitute the API - Application Programming Interface, of *SPPAS*.

4.3. SPPAS 1.4 tools

The main tool to work with SPPAS is *sppas.py* that executes the GUI or that can be used in command-line mode. However, this latter usage does not let the possibility to fix specific options for each module. A set of tools is then available: they propose main programs to deal with the SPPAS API. For each one of these tools, a large set of options can be fixed. SPPAS 1.4 tools are:

wavsplit.py is the main tool to perform IPU-segmentation. This tool enables to find silences from a wav input file, depending on 4 parameters:

- the window width (used to estimate rms) in seconds,
- the minimal speech units duration in seconds,
- the minimal silence duration in seconds,
- the volume in dB.

The silence research can also be controlled by using one of these options: fix a number of units or set a transcription file. This tool can provide three different outputs:

- a directory containing units as a set of wav files and a set of text files with the corresponding transcription,
- a text file containing the list of start time and end time of each unit,
- a TextGrid file with silences/units segmentation.

phonetize.py is the main tool to perform phonetization. This tool takes as input a transcription in the form of a tier in a TextGrid file, and a dictionary. It produces a tier containing the phonetization that optionally can be added to the intput or written in a separate TextGrid file. For units containing unknown words, this tool can optionally use an internal algorithm to phonetize it or it can use the label "UNK".

alignment.py is the main tool to perform alignment. This tool takes as input a phonetization in the form of a tier in a TextGrid file, and an acoustic model. It produces 2 tiers containing the alignments (words and phonemes) that optionally can be added to the intput or written in 2 separate TextGrid files.

LPL-Syllabeur-2.2.jar is the main tool to perform syllabification. This tool takes as input a phoneme alignment in the form of a tier in a TextGrid file, and a configuration file (with a set of rules). It produces a tier containing syllables time-aligned whichnis written in a separate TextGrid file.

momel.py is the SPPAS implementation of the Momel tool. This tool allows to find pitch targets from a wav input file. The Intsint tool can be optionally activated. This tool produces 2 point tiers containing the momel targets and the Intsint labels that can be optionally added to the intput or written in a separate TextGrid file.

SPPAS 1.4 also contains a set of utility tools to deal with wav files, as for example:

- *wavcut.py* is used to cut a wav file by using a start time and a duration.
- *wavstats.py* is used to obtain some statistics about a wav input file.
- *wav2intensity.py* is used to create an intensity tier from a wav input file.

4.4. Evaluations

A large set of evaluations were carried on the phonetization of French (Bigi et al., 2012). Evaluations on Italian were also carried out during the Evalita evaluation campaign: SPPAS participated to the “Forced Alignment on Spontaneous Speech”. Both phonetization and alignment (phonemes and words) were evaluated (Bigi, 2012).

5. Conclusion

SPPAS is a tool to perform automatic phonetic segmentations. *SPPAS* is not only dedicated to computer-scientists. It is rather dedicated to phoneticians because such a tool does not yet exist under GPL license. The only step in the procedure which is probably beyond the means of a linguist without external aid is the creation of a new acoustic model when it does not yet exist for the language being analysed. This only needs to carried out once for each language, though, and we plan to provide detailed specifications of the information needed to train an acoustic model on an appropriate set of recordings and dictionaries or transcriptions. Acoustic models obtained by such a collaborative process will be made freely available to the scientific community.

Version 1.4 deals with TextGrid files. Future development will consist in improving portability to other transcription tools as Transcriber or Elan. Other automatic annotation modules are still in progress: Momel and Intsint are currently implemented. A multilingual tokenizer will also be integrated complying with the architecture proposed in (Bigi, 2011). Finally, we plan to add some languages and to improve the Chinese resources.

6. Acknowledgements

Our thanks to Masahiko Komatsu for allowing us to use the Chinese Multext corpus and to Na Zhi for her help in developing the resources for annotating Chinese.

This work was supported by ANR OTIM project Ref. Nr. ANR-08-BLAN-0239.

7. References

R. Bertrand, P. Blache, R. Espesser, G. Ferré, C. Meunier, B. Priego-Valverde, and S. Rauzy. 2008. Le CID - Corpus of Interactional Data. *Traitement Automatique des Langues*, 49(3):105–134.

B. Bigi, C. Meunier, I. Nesterenko, and R. Bertrand. 2010. Automatic detection of syllable boundaries in spontaneous speech. In *Language Resource and Evaluation Conference*, pages 3285–3292, La Valetta (Malta).

B. Bigi, P. Péri, and R. Bertrand. 2012. Orthographic Transcription: Which Enrichment is required for Phonetization? In *The eighth international conference on Language Resources and Evaluation*, Istanbul (Turkey).

B. Bigi. 2011. A multilingual text normalization approach. In *2nd Less-Resourced Languages workshop, 5th Language & Technology Conference*, Poznàn (Poland).

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

P. Boersma and D. Weenink. 2009. Praat: doing phonetics by computer, http://www.praat.org.

Carnegie Mellon University. 2011. CMUSphinx: Open Source Toolkit For Speech Recognition. http://cmusphinx.sourceforge.net.

J.-Ph. Goldman. 2011. EasyAlign: an automatic phonetic alignment tool under Praat. In *InterSpeech*, Florence (Italy).

S. Herment, A. Loukina, A. Tortel, D. Hirst, and B. Bigi. 2012. A multi-layered learners corpus: automatic annotation. In *4th International conference on corpus linguistics Language, corpora and applications: diversity and change*, Jaen (Spain).

D.-J. Hirst and A. Di Cristo. 1998. *Intonation Systems. A survey of Twenty Languages*.

D.-J. Hirst and R. Espesser. 1993. Automatic modelling of fundamental frequency using a quadratic spline function. *Travaux de l'Institut de Phonétique d'Aix*, 15:75–85.

M. Kipp. 2001. Anvil - a generic annotation tool for multimodal dialogue. In *7th European Conference on Speech Communication and Technology*, pages 1367–1370, Scandinavia.

J. Llisterri. 2011. Speech analysis and transcription software.

Nagoya Institute of Technology. 2010. Open-source large vocabulary csr engine julius, rev. 4.1.5.

H. Sloetjes, A. Russel, and A. Klassmann. 2010. Elan: a free and open source multimedia annotation tool.

The Centre for Speech Technology Research. 2011. The festival speech synthesis system.

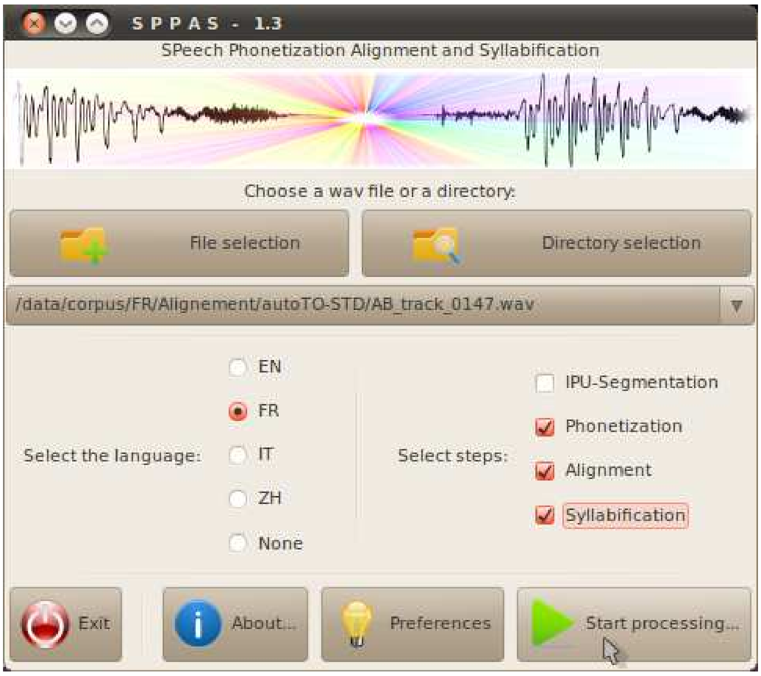
TranscriberAG. 2011. A tool for segmenting, labeling and transcribing speech. [computer software] paris: Dga. http://transag.sourceforge.net/.

VoxForge. 2006-2011. http://www.voxforge.org.

WaveSurfer. 2012. http://www.speech.kth.se/wavesurfer/.

S.J. Young and Sj Young. 1994. The htk hidden markov model toolkit: Design and philosophy. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44.

J. Yuan and M. Liberman. 2008. Speaker identification on the SCOTUS corpus. In *Acoustics*.

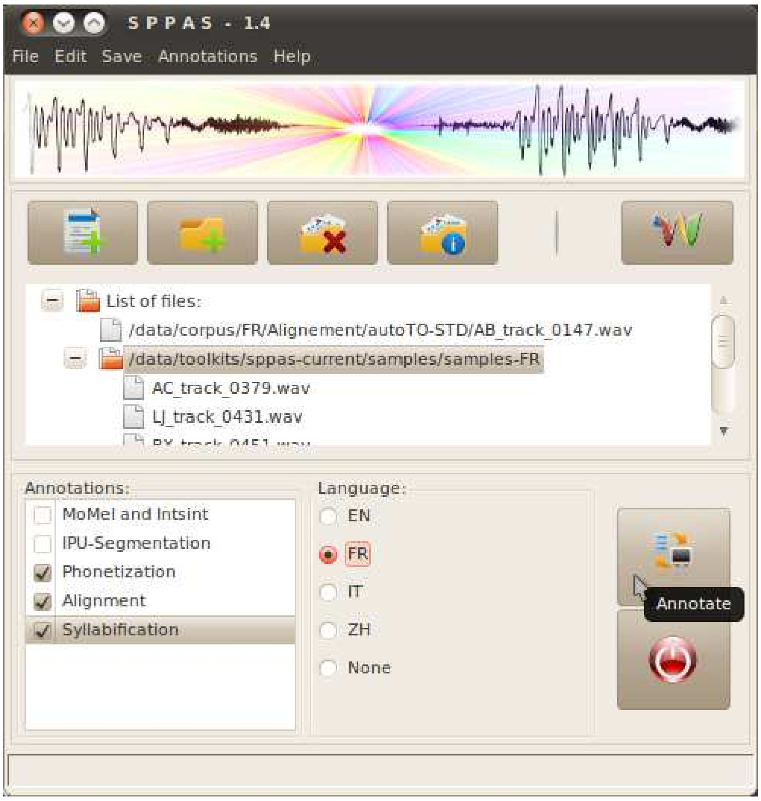


/data/corpus/FR/Alignement/autoTO-STD/AB_track_0147.wav

Select the language: ○ EN ● FR ○ IT ○ ZH ○ None

Select steps: □ IPU-Segmentation ☑ Phonetization ☑ Alignment ☑ Syllabification

Figure 6: SPPAS GUI, version 1.3 (based on pyGtk)



List of files:
- /data/corpus/FR/Alignement/autoTO-STD/AB_track_0147.wav
- /data/toolkits/sppas-current/samples/samples-FR
  - AC_track_0379.wav
  - LJ_track_0431.wav
  - BX_track_0451.wav

Annotations: □ MoMel and Intsint □ IPU-Segmentation ☑ Phonetization ☑ Alignment ☑ Syllabification

Language: ○ EN ● FR ○ IT ○ ZH ○ None

Figure 7: SPPAS GUI, version 1.4 (based on wxpython)