INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Sp1NY: A Quick and Flexible Speech visualisation Tool in Python

Sébastien Le Maguer, Mark Anderson, Naomi Harte

Sigmedia Lab, School of Engineering, Trinity College Dublin, Ireland
{lemagues, andersm3, nharte}@tcd.ie

Abstract

In this submission, we describe Sp1NY, a Python toolkit to visualise and annotate speech. Inspired by Praat and music notation software, we designed Sp1NY to be accessible and flexible. By introducing a control panel, Sp1NY provides a quick way for the user to interact with it. By focusing Sp1NY only on visualisation and annotation and, by reducing the core of the software to a minimum, we ensure that the software will remain stable. Finally, Sp1NY integrates a plugin mechanism which allows researchers to adapt the tool to their needs.

Index Terms: visualisation, Annotation, Speech, Python, Qt, PyQtGraph

1. Introduction

In the era of data analysis and machine learning, visualisation and annotation of data is critical, as are the tools to do so. Not only does the training of good models require well-curated data, but a good visualisation and annotation tool allows researchers to analyse the output of these models. Such a tool is even more essential for speech researchers who manipulate a complex and information-rich signal.

Over the years, multiple tools have been proposed for the visualisation and annotation of speech signals. The most notable tools in use are WaveSurfer [1] and Praat [2]. Both tools exhibit certain strengths. WaveSurfer's strength is its simplicity: the user can visualise and annotate an audio signal with ease, using only a few clicks. Meanwhile Praat is a full annotation and speech processing environment, providing access to an wide array of self-contained features such as speech synthesis and a scripting language.

While these tools are the cornerstones of speech annotation, their development began over 20 years ago. Since then, the fields of User Interface (UI) design, speech technology and software design have evolved significantly. As a result, it is now time to propose a new take on a speech annotation and visualisation tool. In this paper, we present the result of this new take: Speech vIsualization and aNnotation in pYthon (Sp1NY).

Sp1NY is open-source software[1], developed with the following constraints. First, Sp1NY has been designed to focus on speech visualisation and annotation only. This constraint is key to obtaining a simple but efficient tool and avoids introducing unnecessary layers of complexity. Second, the core of Sp1NY has been designed to be as minimal as possible by introducing the usage of visualisation plugins. This ensures that the core of Sp1NY is stable and allows users to adapt Sp1NY for their needs. Finally, the Graphic User Interface (GUI) of Sp1NY has been designed to facilitate the onboarding process and to provide quick access to the annotation and visualisation controls.

We have divided the presentation of Sp1NY into two sections. Section 2 presents the GUI and the design choices which lead to it. Section 3 presents our software design choices and how Sp1NY can be extended to fit the user's needs.

2. User Interface Design

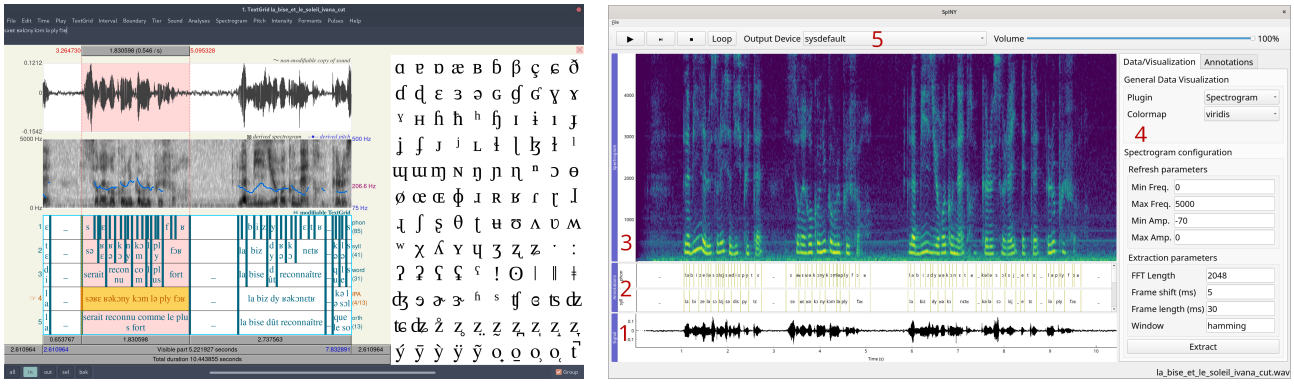
The design of Sp1NY's GUI has drawn inspiration from both Praat and music editing software such as MuseScore 4 [3]. A key strength of Praat is how it handles annotations. From the perspective of the user, all the relevant information is visible and modifiable in one place. This allows for straightforward interaction. However, interaction can become cumbersome when the user wants more refined control over the various aspects of visualisation or annotation. For example, to control the spectrogram rendering, the user has to navigate through different menu items before reaching the relevant controls. Opposing this, the design of music notation software relies on left and right panels, providing quick access to the relevant controls. Music notation software also provide an audio player toolbar, which gives the user a straightforward way to control audio playback. Our goal is to keep the simplicity introduced in Praat to visualise and annotate speech while enhancing the accessibility to the control elements using the panel design scheme from music notation software. To facilitate the presentation of Sp1NY's GUI, we compare it to Praat's GUI as presented in Figure 1.

Sp1NY comprises two main parts: the visualisation area and the control area (4). The visualisation area is similar to Praat (as we can see when comparing Figure 1b to Figure 1a) and is composed of three areas: the waveform viewer (1), the annotation viewer (2) and the speech visualisation viewer (3). Each element of the visualisation part is dockable, this allows the user to move these elements to suit their needs. For example, in Figure 1b, the visualisation used is a spectrogram; however, any visualisation plugin developed for Sp1NY can be used. We discuss plugins further in Section 3.

The control area is divided into two control panels accessible by their respective tabs: the speech visualisation panel and the annotation panel. The visualisation panel allows the user to select which visualisation plugin they want to use as well as define the parameters necessary to control that visualisation. For example, the users can select to visualise the speech using a spectrogram or a wavelet transform (such as [4]) and then define window size/stride, frequency limits and amplitude limits. How plugins are developed is presented in the next section. The annotation panel[2] provides an easy and flexible way to modify

[1] https://github.com/sigmedia/sp1ny

[2] Due to space constraints, we don't include a snapshot of the annotation panel



(a) *Praat* (b) *Sp1NY*

Figure 1: *Comparing the GUIs of Praat (a) and Sp1NY (b). The areas of importance of the GUI of Sp1NY are: 1) the waveform, 2) the annotations, 3) the data visualisation, 4) the control panel and 5) the playback toolbar.*

annotations. It is composed of three parts: the annotation file, the current annotation information and the IPA helpers. The IPA helpers are similar to those used in Praat, but we have improved the user experience through grouping elements by category (i.e. consonants, vowels, diacritics and suprasegmental elements). We also introduce a tooltip, providing a description of the relevant element.

We introduce a flexible way of controlling the annotations using key modifiers. This allows the user to zoom in/out of a segment quickly or to play the portion of the signal associated with that segment in one simple click. The type of operation is defined by the key modifier used.

3. Software Design

Sp1NY is developed in Python. This choice is justified by two main reasons. First, Python is cross-platform, which allows Sp1NY to be available on Linux, MacOS and Windows. Second, multiple speech processing toolkits and packages have been written in Python, enabling developers to extend Sp1NY to suit their needs easily.

As previously mentioned, we designed Sp1NY with a minimal core which is extended through the use of plugins. Sp1NY's core provides the window layout, audio playback, waveform rendering, annotation management and a plugin loading mechanism. Visualisation of a speech representation (e.g. spectrogram or wavelet transform) is delegated to the plugins.

A visualisation plugin is a simple Python package dedicated to instantiating one visualisation paradigm. The plugin package has to be a subpackage of `spiny.plugins`. This enables us to automatically load the plugin if it is in the python path. As illustrated in Figure 2, a plugin is composed of three classes: the Extractor, which extracts the representation from the speech signal (method `extract`); the View, which renders of the representation obtained by the Extractor (method `refresh`); and the Controller, which calls the Extractor and provides the parameters necessary for the representation extraction (method `update`, which internally needs to call the methods `extract` from the Extractor and `refresh` from the View).

Both the Controller and the View require visual components, allowing the user to interact with them. For visual rendering, Sp1NY relies on two libraries: PyQt and PyQtGraph[5]. PyQt is a set of Python bindings for Qt, which is a standard, cross-platform GUI library used in software such as MuseScore

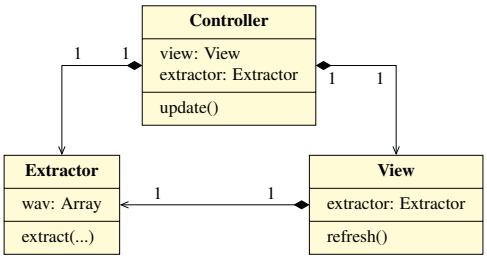


Figure 2: *Class diagram describing the architecture of a Sp1NY plugin.*

4. PyQtGraph is a library built on top of PyQt which provides a fast way to plot and visualise data. As a result, Sp1NY leverages PyQtGraph to render the data visualisation, the annotations and the waveform and uses PyQt to design any other GUI components. Considering a plugin, the View is developed using PyQtGraph (more precisely, the View is a subclass of `pg.PlotWidget`), while the GUI of the Controller is developed using PyQt. An advantage of using PyQt is that it provides an easy way to design elements of the GUI using QtDesigner and Qml. This is especially convenient as it allows the plugin developer to prototype the control panel quickly.

4. References

[1] K. Sjölander and J. Beskow, "Wavesurfer - an open source speech tool," in *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, 2000, pp. vol. 4, 464–467.

[2] P. Boersma and D. Weenink, "Praat, a system for doing phonetics by computer, version 3.4," *Institute of Phonetic sciences of the University of Amsterdam, Report*, vol. 132, p. 182, 1996.

[3] "Musescore 4," https://musescore.org/en/4.0, Dec. 2022.

[4] A. Suni, J. Šimko, D. Aalto, and M. Vainio, "Hierarchical representation and estimation of prosody using continuous wavelet transform," *Computer Speech & Language*, vol. 45, pp. 123–136, 2017.

[5] "Pyqtgraph - scientific graphics and gui library for python," https://www.pyqtgraph.org/.

[6] H. N. Bicer, P. Gotz, C. Tuna, and E. A. P. Habets, "Explainable acoustic scene classification: Making decisions audible," in *International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, Sep 2022. [Online]. Available: http://dx.doi.org/10.1109/iwaenc53105.2022.9914699