Vrije Universiteit Brussel

Demonstrating Parselmouth: Integrating Praat into a complex Python workflow

Jadoul, Yannick

Publication date:
2018

Link to publication

Citation for published version (APA):
Jadoul, Y. (2018). *Demonstrating Parselmouth: Integrating Praat into a complex Python workflow*. Poster session presented at Interdisciplinary Approaches in the Language Sciences: International Max Planck Research School for the Language Sciences Conference 2018, Nijmegen, Netherlands.

Copyright
No part of this publication may be reproduced or transmitted in any form, without the prior written permission of the author(s) or other rights holders to whom publication rights have been transferred, unless permitted by a license attached to the publication (a Creative Commons license or other), or unless exceptions to copyright law apply.

Take down policy
If you believe that this document infringes your copyright or other rights, please contact openaccess@vub.be, with details of the nature of the infringement. We will investigate the claim and if justified, we will take the appropriate steps.

Download date: 18. Jan. 2025

Demonstrating Parselmouth: Integrating Praat into a complex Python workflow

Yannick Jadoul, Bill Thompson, Bart de Boer – Artificial Intelligence Lab, Vrije Universiteit Brussel
Yannick.Jadoul@ai.vub.ac.be

Parselmouth

Interdisciplinary science requires interdisciplinary tools: as different scientific disciplines combine methods and approaches to research language and speech, existing algorithms and tools need to be combined. One of these tools is Praat (Boersma and Weenink, 2018), a software package implementing a wide range of acoustic and phonetic algorithms and analyses. While Python and other scripting languages often allow different parts of the research to be automated and glued together into a single workflow, doing so with Praat and its scripting language is not always as straightforward. To simplify the integration of Praat into a complex workflow, we have developed Parselmouth, a Python interface to Praat.

Why and when use Parselmouth?

- Complex workflows (vs. interactive views in Praat)
- Interactive and adaptive use (vs. preprocessing data)
- Integration with other Python libraries (e.g. NumPy)
- Custom control over data, results, or visualization

Installation and usage

```
$ pip install praat-parselmouth

>>> import parselmouth
```

Parselmouth @ GitHub
https://github.com/YannickJadoul/Parselmouth

Web service example @ ReadTheDocs
https://parselmouth.readthedocs.io/en/latest/examples/web_service.html

Plotting example @ ReadTheDocs
https://parselmouth.readthedocs.io/en/latest/examples/plotting.html

PsychoPy experiments example @ ReadTheDocs
http://parselmouth.readthedocs.io/en/latest/examples/psychopy_experiments.html

Python web server

- Web experiment, e.g. Amazon Mechanical Turk
- Praat functionality on web server
- Participant/client side independent from Praat

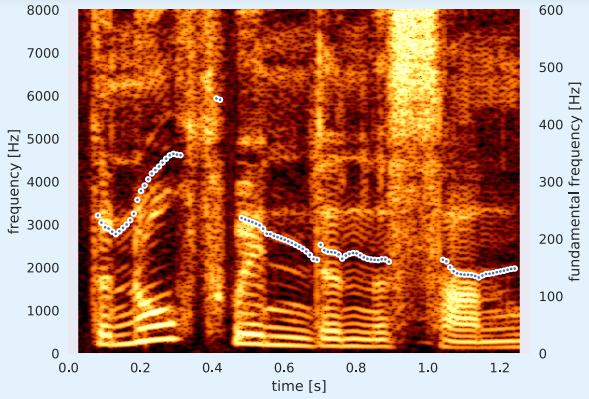
Python server

```
from flask import Flask, request, jsonify
import tempfile

app = Flask(__name__)

@app.route('/pitch_track', methods=['POST'])
def pitch_track():
    import parselmouth
    with tempfile.NamedTemporaryFile() as tmp:
        tmp.write(request.files['audio'].read())
        sound = parselmouth.Sound(tmp.name)
    return jsonify(list(sound.to_pitch().selected_array['frequency']))
```

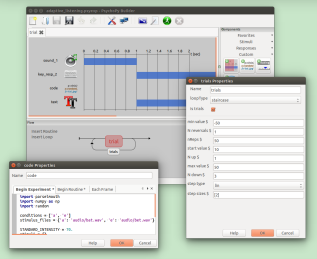
Sample client

```
import requests
import json

files = {'audio': open('the_north_wind_and_the_sun.wav', 'rb')}
reply = requests.post('http://127.0.0.1:5000/pitch_track', files=files)
print(json.loads(reply.text))
```

Custom visualizations in Python



Adaptive PsychoPy experiments



- Experimental framework in Python with graphical user interface
- Insert custom Python code
- Access Praat functionality with Parselmouth
- Generate stimuli on the fly
- Adapt stimuli to responses

Before Experiment

```
import parselmouth
import numpy as np
import random

conditions = ['a', 'e']
stimulus_files = {'a': 'audio/bat.wav', 'e': 'audio/bet.wav'}

STANDARD_INTENSITY = 70.
stimuli = {}
for condition in conditions:
    stimulus = parselmouth.Sound(stimulus_files[condition])
    stimulus.scale_intensity(STANDARD_INTENSITY)
    stimuli[condition] = stimulus
```

Before Routine

```
random_condition = random.choice(conditions)
random_stimulus = stimuli[random_condition]

noise_samples = np.random.normal(size=random_stimulus.n_samples)
noisy_stimulus = parselmouth.Sound(noise_samples,
                    sampling_frequency=random_stimulus.sampling_frequency)
noisy_stimulus.scale_intensity(STANDARD_INTENSITY - level)
noisy_stimulus.values += random_stimulus.values
noisy_stimulus.scale_intensity(STANDARD_INTENSITY)

# 'filename' variable is set by PsychoPy and contains base file name
# of saved log/output files, so we'll use that to save our custom stimuli
stimulus_file_name = filename + '_stimulus_' + str(trials.thisTrialN) + '.wav'
noisy_stimulus.resample(44100).save(stimulus_file_name, "WAV")
sound_1.setSound(stimulus_file_name)
```

After Routine

```
trials.addResponse(key_resp_2.keys == random_condition)
```

Demonstrating Parselmouth: Integrating Praat into a complex Python workflow

Yannick Jadoul[a,b,*], Bill Thompson[c,d,a,**], and Bart de Boer[a,d,***]

[*] Yannick.Jadoul@ai.vub.ac.be
[**] biltho@mpi.nl
[***] bart@ai.vub.ac.be

[a] Artificial Intelligence Lab Brussels, Vrije Universiteit Brussel
[b] ADReM Research Group, University of Antwerp
[c] Department of Psychology, University of California, Berkeley
[d] Language and Cognition Department, Max Planck Institute for Psycholinguistics

Interdisciplinary science requires interdisciplinary tools: as different scientific disciplines combine methods and approaches to research language and speech, existing algorithms and tools need to be combined. One of these tools is Praat (Boersma and Weenink, 2018), a software package implementing a wide range of acoustic and phonetic algorithms and analyses. While Python and other scripting languages are designed such that they allow automation and gluing together of different parts of the research into a single workflow, doing so with Praat and its scripting language is not always as straightforward. To simplify the integration of Praat into a complex workflow, we have developed Parselmouth.

Parselmouth is an open-source Python library[1,2] that allows one to access Praat functionality and combine it with other Python tools and libraries, yet feels natural and simple to a Python user. As such, we believe Parselmouth can be useful in a variety of cases, ranging from the visualization of acoustic data or batch analysis of audio files, to computational models on speech perception and acquisition.

We also introduce the possibility of integrating Praat's phonetic analysis and manipulation of responses and stimuli in an adaptive experiment, using Parselmouth. One widely-used Python framework for setting up and running *"a wide range of neuroscience, psychology and psychophysics experiments"* is PsychoPy (Peirce, 2007), which includes a graphical interface to build experiments with a minimal amount of coding.

We demonstrate the integration of Parselmouth into PsychoPy experiments with an adaptive staircase experiment (e.g., Kaernbach, 2001; de Boer, 2012) to determine the minimal amount of noise that stops participants from correctly classifying a stimulus. Using Praat functionality through Parselmouth, at each step *during the experiment* a new audio stimulus is created with the desired signal-to-noise ratio. With the presented examples, we hope to illustrate the new, modern experimental setups and workflows that Parselmouth facilitates, advancing interdisciplinary research to answer questions in language sciences.

[1] https://github.com/YannickJadoul/Parselmouth
[2] https://parselmouth.readthedocs.io/en/latest/

References

Boersma, P. and Weenink, D. (2018). Praat: doing phonetics by computer [Computer program]. Version 6.0.37, retrieved 3 February 2018 from http://www.praat.org/.

de Boer, B. (2012). Loss of air sacs improved hominin speech abilities. *Journal of human evolution*, 62(1):1–6.

Kaernbach, C. (2001). Adaptive threshold estimation with unforced-choice tasks. *Attention, Perception, & Psychophysics*, 63(8):1377–1388.

Peirce, J. W. (2007). PsychoPy – psychophysics software in Python. *Journal of neuroscience methods*, 162(1-2):8–13.