PyProsProp Release Notes

Anindita Nath, Alonso Grandos, Nigel Ward. University of Texas at El Paso

February 28, 2018

PyProsProp is the python version of ProsProp, a code suite for inferring various semantic- and pragmatic-level properties of an audio file from its prosody.

Use Cases: 1) Inferring stance, as described in the Computer Speech and Language journal article: http://www.cs.utep.edu/nigel/papers/csl-stance.pdf . 2) Inferring situation frame types in the Lorelei context.

Documentation: The full documentation for ProsProp's is available at its github repository: https://github.com/nigelgward/stance . This document only notes the differences in the python version.

Provenance: This version is a fairly literal translation of the Matlab code into Python 3.6. The major differences are in the use of a different pitch tracker and the use of a different cepstrum algorithm. It therefore does not perform exactly the same. This release is non-modular, combining in one director the prosodic features computations ("*midlevel*") and inference processes ("stance") that are separate packages in the Matlab version. This release is minimalistic, including only the essentials for the workflow, leaving out many ancillary functions for analysis and tuning that were present in the Matlab.

Caveat: This is research code, designed to support experimentation. We have taken care to structure it so that it can be adapted for production use, but in its current state it is not truly robust.

Testing so far:

i. This code has been tested on English, Mandarin and Turkish audios, in a leave-one-out regime. The corresponding models and prediction outputs are delivered.

ii. The performance of the code is compared with the Matlab version and the mean squared error of the overall predictions corresponding to the input language are enumerated in the following table. This code is much slower than its Matlab equivalent.

Language	**Python**		**Matlab**	
	Baseline	Model	Baseline	Model
English	0.43	0.48	0.43	0.45
Mandarin	0.43	0.30	0.43	0.22
Turkish	0.25	0.22	0.25	0.25

As the table shows, overall MSE of predictions generated by Matlab is slightly better than that of Python for all languages other than Turkish.

Planned Extensions and Improvements:

i. Create a model for situation-frame inference for one of the Lorelei IL languages. This will also require new functions to deal with the different file formats.

ii. Further clean up and optimize the code.

Restrictions on Use: None. This distribution is mostly code written at UTEP, which we release without restriction for use by any one for any purpose. It also includes code by James Lyon (base.py) and by Bernardo J. B. Schmitt (pYAAPT.py etc. in amfm), which we believe also to be in the public domain. Further it includes as test audios news broadcasts downloaded from archive.org, which we believe also has no restrictions on use or redistribution.

Steps to download and test:

i. Start up *Python* (version 3.6.0 or higher). (For example, download Anaconda 3, run it, and launch a Spyder console; the project has been created in this.)

ii. Dependencies (need to be installed manually if they do not exist):
    a) *h5py* version 2.7.1, *hdf5* version 1.10.10.
    b) *numpy* version 1.13.3.
    c) *scikit-learn* version 0.19.1.
    d) *scipy* version 1.0.0.

iii. From the menus, use File->Open to open *src/combinedAPI.py.*

iv. Line #17: the language of the audio is entered as a user input in an abbreviated form.

    language =input("Enter the Language Abbreviation, 'E' for English, 'M' for Mandarin and 'T' for Turkish: "); for any other language, the annotation format are supposed to similar to that of *'English'* (default format).

v. Line # 30- line #33 : these are various running instances corresponding to different languages which will create the model, save it as a .mat file in the *src* directory by calling :

    makePPM.py ('path to the audio directory', 'path to the annotations directory', 'path to the featurefile (.fssfile)', 'name of the ppmfile', 'abbreviated name of the language of the audio')

vi. Line # 46- line #50 : these are various running instances corresponding to different languages which run the test and save the predictions as another .mat file in the

output_python [*languageAbbreviation*] directory (auto created if it does not exist) in the *src* directory, and also save the predictions as *pypredictions*[*languageAbbreviation*].*txt*.
by calling :

regressionTest.py (‘path to the audio directory’, ‘path to the annotations directory’, ‘name/path to the ppmfile created above’, ‘abbreviated name of the language of the audio’)

vii. Also, the midlevel features are saved as a .mat file in *monster[language abbreviation]* under the same directory as in (*vii*).

viii. All the corresponding standard output while running the *makePPM.py* and *regressionTest.py* are saved in 2 text files namely, *ppm*[*language abbreviation].txt and evaluate[language abbreviation].txt,* respectively under the same directory as in (*vii*).The profiling information, generated in 2 ways (one which includes information about parent and child functions) saved in 2 text files namely, *profile[language abbreviation]_callers.txt* and *profile[language abbreviation].txt* under the same directory as in (*vii*).

Alternative Top-Level Ways to Run the Code:

i. To use the command line interface, open the command prompt from within the source-code directory(*src*) and run:

python maininterface.py -audio *[vii.a]* -annotations *[vii.b]* -featurefile *[vii.c]* -ppmfile *[vii.d]* -langAbb *[vii.e]*

ii. To create the model and test in separate steps, use *makePPM.py* and *regressionTest.py.*

New API: A new API is created to predict stances from the audios when no annotations are provided with the *ppmmodel* of any language provided as input.

Ways to run the new API code:

i. Open the script *new_noAnno_API.py* and uncomment any one of the lines (line #28 - line #31) under heading *“test”* to generate run instances as described below:

a) *noAnnoAPI*('path to the test audio directory', 'path to the ppmmodel', ‘name of the output file', ‘audio-language’ , ‘start of the audio segment in secs’ , ‘end of the audio segment in secs’) ##### all optional arguments are mentioned

b) *noAnnoAPI* ('path to the test audio directory', 'path to the ppmmodel', ‘name of the output file', ‘audio-language’) ##### 0 optional argument, entire audio is treated as a single segment

c) *noAnnoAPI* ('path to the test audio directory', 'path to the ppmmodel', ‘name of the output file', ‘audio-language’ , ‘end of the audio segment in secs’) ##### 1 optional argument : audio segment from the start till the ‘*end’* second of the audio

d) *noAnnoAPI* ('path to the test audio directory', 'path to the ppmmodel', ‘name of the output file', ‘audio-language’ , ‘start of the audio segment in secs’) ##### 1 optional argument: audio segment from the *‘start’* secs of the audio till its end

ii. A *ppmmodel* is provided in *‘test_ppmmodel_English/ ppmaudiosengpy.mat’*.
iii. A test audio is provided in *‘testAudio’.*

Note: You can create your own *ppmmodel* by uncommenting the first line under the same heading in the above script which calls *makePPM.py*.

Contact: nath.anindita2110@gmail.com