PyProsProp Release Notes

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

December 20, 2017

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. 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.

|  |  |  |
| --- | --- | --- |
| **Stance** | **Matlab** | **Python** |
| 1 | 0.41 | 0.37 |
| 2 | 0.59 | 0.61 |
| 3 | 0.03 | 0.03 |
| 4 | 0.36 | 0.38 |
| 5 | 0.18 | 0.18 |
| 6 | 0.04 | 0.05 |
| 7 | 0.20 | 0.19 |
| 8 | 0.06 | 0.05 |
| 9 | 0.98 | 0.79 |
| 10 | 0.12 | 0.11 |
| 11 | 0.54 | 0.51 |
| 12 | 0.38 | 0.33 |
| 13 | 0.54 | 0.47 |
| 14 | 0.42 | 0.42 |
| **average** | **0.35** | **0.32** |

**Testing so far**: This code has been tested on 19 news segments in a 10-minute English radio broadcast, in a leave-one-out regime. As the table shows, for most stances the mean squared error of the Python-generated predictions is slightly lower than that for the Matlab-generated predictions.

**Planned Extensions and Improvements:** 1) Create and deliver models for stance inference for English, Mandarin, and Turkish using large amounts of data. 2) Compare performance to that of the Matlab code. 3) 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. 4) Speed things up. Currently the Python is much slower than the Matlab. We suspect that the main time sinks are file I/O and data structure conversion (reshaping).

**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 audio a news broadcast downloaded from archive.org, which we believe also has no restrictions on use or redistribution.

**Steps to download and test:**

1. Download the code and test data (21 MB) from

<http://www.cs.utep.edu/nigel/for-tina.zip>

1. Unzip it to some convenient directory.
2. Start up Python (version 3.6.0 or higher). (For example, download Anaconda 3, run it, and launch a Spyder console. Need to install numpy and scipy if they are not auto-installed.)
3. From the menus, use File->Open to open src/regressionTest.py.
4. Run regressionTest. This will create the model, save it as a .mat file in the src directory, run the test and save the predictions as another .mat file in the outputs\_python directory (auto creating it if it does not exist) under the src directory, and also save the predictions as pypredictions.txt. This will take about two minutes.
5. Compare the newly created pypredictions.txt to the version in testeng: they should match exactly.

**Alternative Top-Level Ways to Run the Code**:

1. To use the command line, open the command prompt from within the source-code directory and run:

python maininterface.py -audio *[iv.a]* -annotations *[iv.b]* -featurefile *[iv.c]* -modelfile *[iv.d]*

1. To create the model and test in in separate steps, use makePPM.py and prosprop.py.

**Contact:** nath.anindita2110@gmail.com