

Prosody Toolkit: Integrating HTK, Praat and WEKA

S. Thomas Christie¹, Serguei Pakhomov²

¹Cognitive Science, University of Minnesota

²Center for Clinical and Cognitive Neuropharmacology, College of Pharmacy, University of Minnesota

tchristie@umn.edu, pakh0002@umn.edu

Abstract

A major hurdle in computational speech analysis is the effective integration of available tools originally developed for purposes unrelated to each other. We present a Python-based tool to enable an efficient and organized processing workflow incorporating automatic speech recognition using HTK, phoneme-level prosodic feature extraction in Praat and machine learning in WEKA. Our system is extensible, customizable and organizes prosodic data by phoneme and time stamp in a tabular fashion in preparation for analysis using other utilities. Plotting of prosodic information is supported to enable visualization of prosodic features.

Index Terms: prosody, Python, HTK, Praat, WEKA

1. Introduction

The speech community has access to high quality and established tools for the computerized analysis of recorded speech. The Hidden Markov Model Toolkit (HTK) [1] is widely used for word and phoneme recognition, as is Praat [2] for prosodic analysis. We present a platform-independent solution for incorporating information from the two programs and preparing it for easy importing into applications such as MATLAB and R, as well as automatic Attribute-Relation File Format (ARFF) file compilation in preparation for phoneme classification or clustering in WEKA [3], a data mining and machine learning tool.

The Prosody Toolkit (ProTK) is written to be compatible with Python version 2.6 [4] in order to utilize the NumPy [5] and Matplotlib [6] libraries. The toolkit was written for use with Mac OS X, but operates in Linux and Windows environments with only minor modifications. ProTK was originally created to enable detection of filled pauses (um's and ah's) by automatically classifying HTK phoneme level output; however, it may be used for other speech analysis purposes as well.

The development of ProTK was inspired by ProsodyPro [7], a tool for leveraging Praat for prosodic feature extraction and organization, and work by Elizabeth Shriberg [8] in non-lexical, prosody-only detection of disfluencies in spontaneous speech. ProTK is publicly available at <http://rxinformatics.umn.edu>.

2. Capabilities

The Prosody Toolkit is designed for batch prosodic feature extraction from a set of recorded audio. The toolkit consists of a set of Python scripts, a configuration file and a customizable Praat script. ProTK can be run in fully automated mode from the command line, or interactively from within the Python interpreter. The primary output of the program is a directory structure containing phoneme-level prosodic data in a tabular format. As an option, the program can output an ARFF file for phoneme-level classification using WEKA, as discussed below.

ProTK also supports parallel processing and data extraction, so computations are scalable to multiple processors and cores. Figure 1 illustrates a typical workflow.

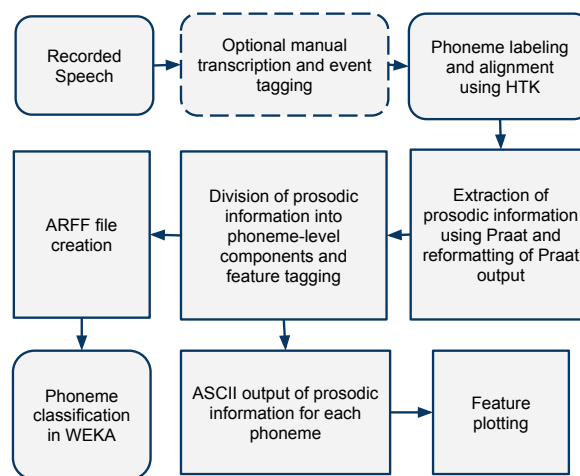


Figure 1: A typical ProTK workflow. Ovals represent processes and data external to ProTK, and rectangles represent processes within ProTK.

2.1. Extraction of prosodic features using Praat

Praat is freely available, cross-platform software for prosodic feature extraction and analysis of audio data. Praat commands are fully scriptable using a customized scripting language, and scripts can be called from the command line. The program also supports the extraction of an array of prosodic features and has a built-in mechanism for visual display of the results. However, ASCII output from Praat is in a difficult-to-parse format that is not easily amenable for input to other programs such as MATLAB or WEKA.

For each file in the batch to be processed, ProTK first calls Praat feature extraction routines. The user can select which features are extracted by editing the included configuration file. Currently supported features include pitch, intensity, formants and their ratios, MFCC and LPC coefficients, jitter and shimmer values, point process values, periods of silence and raw amplitude. Praat routines are called with default parameters, though parameters are editable from within the toolkit. The Praat output for each feature is parsed and reformatted into a tabular format with headings, and subfolders are created to contain file-level prosodic information.

2.2. Organization by phoneme

For our goal of filled pause detection, we needed to segregate and summarize prosodic information associated with each phoneme. After using HTK for phoneme labeling and alignment, ProTK converts the output to a Praat-compatible TextGrid format. Once prosodic data for a file is extracted and reformatted, ProTK uses the TextGrid files to locate and extract prosodic information for each phoneme, optionally including adjacent contextual information, and stores the data in tabular format. Each sub-file is stamped with the phoneme, feature type and time, and organized in a folder structure.

2.3. Data persistence and plotting

In addition to textual storage of prosodic data, ProTK keeps a persistent record of the project and associated file information using Python's Pickle module. After initial data extraction, a project can be opened interactively from the Python terminal, where a user can examine information associated with individual files or phonemes. In addition, the mean and standard deviation for every phoneme-feature combination is stored and used to create ARFF files for subsequent phoneme classification.

The toolkit supports plotting of extracted prosodic information using the Matplotlib library for Python. Users can plot any prosodic information associated with the duration of the phoneme, and information from adjacent time periods can be included for context. The phoneme or tagged feature can be highlighted in the image as shown in Figure 2.

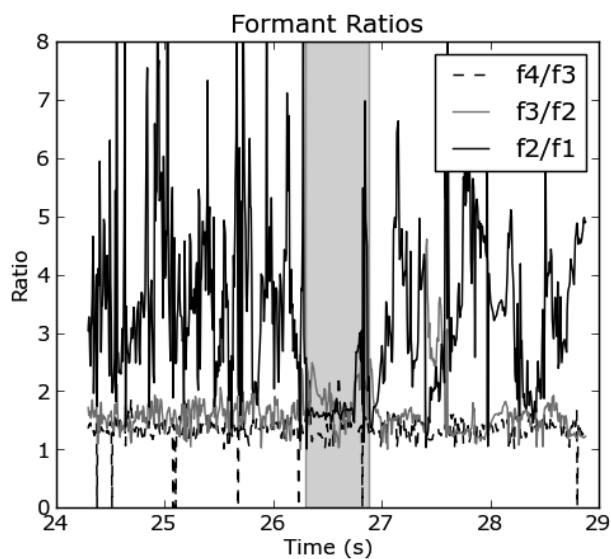


Figure 2: Example plot of formant ratios. Shaded area indicates a filled pause.

2.4. Feature Tagging and WEKA

The toolkit was originally designed for tagging filled pauses in audio files for the purpose of training a classifier. In order to use this feature, the user must provide a TextGrid file with word-level labels (including filled pause labels), as well as phoneme labels and alignments (detected using forced-alignment or in a fully automated mode). When extracting prosodic information associated with each phoneme, ProTK tags each phoneme by default as a Filled Pause or Not Filled Pause, depending on

whether the phoneme's center occurs during a filled pause in the manually labeled TextGrid file. Support for tagging of other dysfluency events (e.g., false starts, repairs, repetitions) as well as other prosodically conditioned events (e.g., utterance boundaries, turn switches, etc.) and user-defined events is currently being developed.

Once prosodic information is extracted and each phoneme is appropriately tagged, ProTK creates an ARFF file for each data set that can be opened by WEKA. In addition to mean and standard deviation values for each phoneme, the normalized duration and label of each phoneme is included, as well as the distance from nearest silence and, optionally, prosodic information associated with adjacent phonemes.

2.5. Command Line Options

The command line usage of ProTK supports several arguments for modifying the processing workflow. The extraction of prosodic information using Praat is optional, as is separating prosodic information into phoneme-level text files. This is useful for saving disk space, and ensures that time-consuming prosodic feature extraction only needs to be run once for each project. The number of processor cores to be used can also be specified. Processing is parallelized at the level of individual recordings, so a separate process is launched for the extraction of information from each file up to the number specified.

Any specified file can be used as the source for tagging information, and multiple sets of files can be processed simultaneously (for evaluating different phoneme alignment settings, for example).

3. Conclusion and Future Work

The current state of the Prosody Toolkit represents an initial exploration into the production, organization and visualization of prosodic data at the phonemic level. The toolkit is under active development and options for increased capability and usability are being considered. Work is currently being done to support additional and user-specifiable feature tags, as well as a graphical user interface for better ease of use.

4. References

- [1] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, and D. Povey, "The HTK book (for HTK version 3.4)," *Cambridge University Engineering Department*, vol. 2, no. 2, pp. 2–3, 2006.
- [2] P. Boersma and D. Weenink, "Praat: Doing phonetics by computer, version 4.0.26," Retrieved August 6, 2010, from <http://www.praat.org>, 2002.
- [3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.
- [4] G. Van Rossum and C. v. W. e. Informatica, *Python reference manual*. Centrum voor Wiskunde en Informatica, 1995.
- [5] T. E. Oliphant, *A Guide to NumPy*. Trelgol Publishing, 2006, vol. 1.
- [6] J. D. Hunter, *Matplotlib: A 2D Graphics Environment*, 3 2009.
- [7] Z. Huang, L. Chen, and M. Harper, "An open source prosodic feature extraction tool," in *Proceedings of the conference on language resources and evaluations (LREC)*, 2006.
- [8] E. Shriberg, R. Bates, and A. Stolcke, "A prosody only decision-tree model for disfluency detection," in *Fifth European Conference on Speech Communication and Technology*, 1997.