Purdue Prosodic Feature Extraction Tool on Praat

Zhongqiang Huang, Lei Chen, Mary P. Harper
Spoken Language Processing Lab
School of Electrical and Computer Engineering
Purdue University, West Lafayette

June 23, 2006

Contents

1 Implementation using Praat **4**
- 1.1 Audio and Word and Phone Alignment . . . 6
- 1.2 Vowel and Rhyme . . . 6
- 1.3 VUV, Raw and Stylized Pitch, and Pitch Slope . . . 6
- 1.4 Raw and Stylized Energy, and the Energy Slope . . . 7
- 1.5 Statistics . . . 7

2 Using the Tool **8**
- 2.1 Global Statistics Computation . . . 8
- 2.2 Prosodic Feature Extraction . . . 9

3 Architecture of the Tool and Its Potential Augmentation **12**
- 3.1 Structure . . . 12
- 3.2 Code Organization . . . 14
- 3.3 An Augmentation Example . . . 15

A Prosodic Feature List **16**
- A.1 Introduction . . . 16
- A.2 Basic Features . . . 16
  - A.2.1 Base Features . . . 16
  - A.2.2 Duration Features . . . 17
  - A.2.3 F_0 Features . . . 18
  - A.2.4 Energy Features . . . 21
- A.3 Statistical Tables . . . 23
- A.4 Derived Features . . . 24
  - A.4.1 Normalized Word Duration . . . 24
  - A.4.2 Normalized Pause . . . 25
  - A.4.3 Normalized Vowel Duration . . . 25
  - A.4.4 Normalized Rhyme Duration . . . 25
  - A.4.5 F_0 Derived Features . . . 26
  - A.4.6 Energy Derived Features . . . 29
  - A.4.7 Average Phone Duration . . . 31
  - A.4.8 Speaker Specific Normalization . . . 31

Version history

- version 0.1
    1. The first public release version.

Introduction

The prosody (i.e., the duration, pitch, and energy) of speech plays an important role in human communication. Research in speech and language processing has shown that the prosodic content of speech can be quite valuable for accurate event tagging. Prosodic cues have been exploited in a variety of spoken language processing tasks such as sentence segmentation and tagging [7, 9], disfluency detection [8], dialog act segmentation and tagging [1], and speaker recognition [12], using the "direct modeling" approach [11]. An advantage of this approach is that no hand segmentation or intermediate labeling of the prosodic content is required (although if it were available it could be used). Instead the prosodic features are extracted directly from the speech signal given its time alignment to a human generated transcription or to automatic speech recognition (ASR) output. A prosody model can then be trained using these features and combined with a language model to build an event detection system.

Many of the past efforts on speech event detection utilize simple prosodic features such as pause duration [4]. By contrast, the above direct modeling efforts utilize a large number of features extracted using a proprietary prosodic feature extraction suite developed at SRI [3] to good effect. SRI's feature extraction tool is Unix script-based, combining *ESPS/Waves* for basic prosodic analysis (e.g., preliminary pitch tracking and energy computation (*get_F0*)) with additional software components, such as a piecewise linear model [12] for pitch stylization.

We have developed an open source automatic prosodic feature extraction tool[1] [5] based on *Praat* [2] to extract a wide variety of prosodic features for event detection tasks that was inspired by the SRI suite. By creating this tool, we hope to provide a framework for building stronger baseline comparisons among systems and to support more effective sharing of prosodic features.

This document is organized as follows: Chapter 1 discusses some important implementation details. Chapter 2 is a user manual on the tool. Chapter 3 gives the structure of the tool as well as an augmentation example that demonstrates the procedures to modify the code in order to extract additional features. An exhaustive list of all of the prosodic features implemented in our tool is given in Appendix A.

[1]This tool can be downloaded at `ftp://ftp.ecn.purdue.edu/harper/praat-prosody.tar.gz` along with a manual [6].

Chapter 1

Implementation using Praat

This chapter is mainly about how *Praat*'s data structures and functionality support prosodic feature extraction. The tool described in this document was designed to extract a set of prosodic features given an audio file and its corresponding word and phone alignments. It is assumed that the alignments are in *Praat TextGrid* format, as in Figure 1.1.

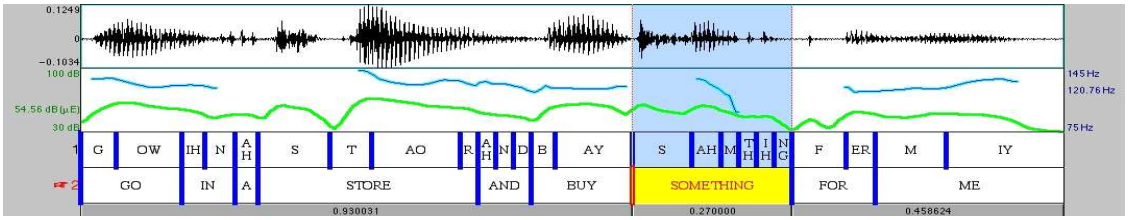

Figure 1.1: An example of a word/phone alignment in *Praat TextGrid* format, together with the waveform.

Given a corpus with audio and time aligned words and phones as input, our tool first extracts a set of basic elements (e.g., raw pitch, stylized pitch, voiced/unvoiced segmentation (VUV)) representing duration, F_0, and energy information, using the procedures illustrated in Figure 1.2. Then a set of duration statistics (e.g., the means and variances of pause duration, phone duration, and last rhyme duration), F_0 related statistics (e.g., the mean and variance of logarithmic F_0 values), and energy related statistics are calculated. Given the duration, F_0, and energy information, as well as the statistics, it is straightforward to extract the prosodic features at each word boundary, according to the definition of features in Appendix A (see also Figure 3.1 for the data flow diagram of the tool). Table 1.1 summarizes the use of raw duration, F_0, and energy in the computation of the prosodic features.

In the rest of this chapter, we describe the requirements needed to use the tool, in particular we discuss audio file and word/phone alignment requirements. We also give details on how vowel, rhyme, VUV, pitch (raw, stylized, and its slope), and energy (raw, stylized, and its slope) are calculated.

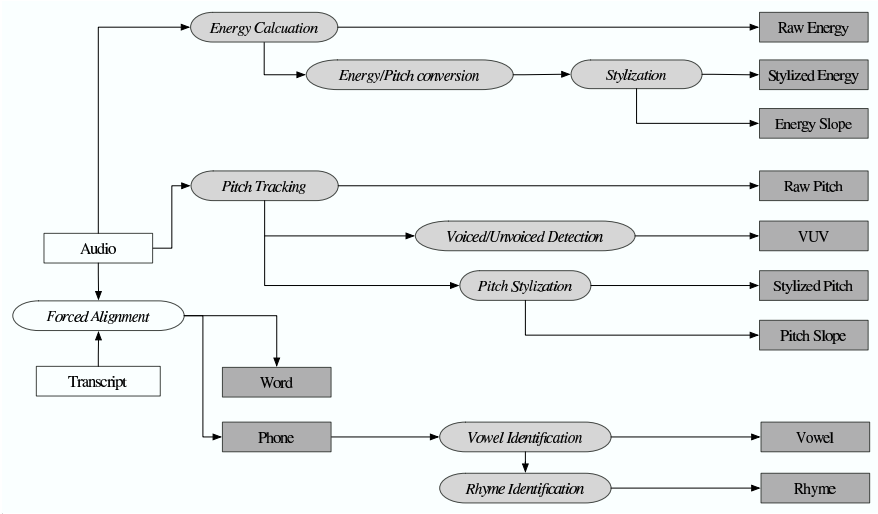

Figure 1.2: Procedures to obtain the basic elements that are directly needed for prosodic feature extraction. The grayed ovals represent operations implemented in the tool, while the grayed rectangles represent the basic elements. Note that Forced Alignment is not a part of the tool, and so it appears in white.

Table 1.1: The use of basic elements for extracting various features. For example, the word alignment is used to compute duration features, F_0 features, and energy features, while the voiced/unvoiced segmentation (VUV) is only used to compute F_0 features.

	Duration Features	**F_0 Features**	**Energy Features**
Word	$\surd$	$\surd$	$\surd$
Phone	$\surd$	$\times$	$\times$
Vowel	$\surd$	$\times$	$\times$
Rhyme	$\surd$	$\times$	$\times$
VUV	$\times$	$\surd$	$\times$
Raw Pitch	$\times$	$\surd$	$\times$
Stylized Pitch	$\times$	$\surd$	$\times$
Pitch Slope	$\times$	$\surd$	$\times$
Raw Energy	$\times$	$\times$	$\surd$
Stylized Energy	$\times$	$\times$	$\surd$
Energy Slope	$\times$	$\times$	$\surd$

1.1 Audio and Word and Phone Alignment

An audio file can be any format (e.g., WAV and AIFF files.) that can be loaded in *Praat* entirely using the "Read from file..." command. If the audio quality is poor, the quality of the pitch and energy features may be degraded.

Word and phone alignments are required to be in *Praat TextGrid* format, one tier each in a separate file. Silence intervals should be empty. We also require that the timing of the phones should align with the timing of the word at the start and end of a word boundary. Since the prosodic features are extracted around the word boundaries, it is important to have high quality alignments[1]. Different pronunciation dictionaries may use different phoneme sets, however, in our system currently we assume that the labels from the phone alignment should be consistent with CMU's dictionary or ISIP's dictionary, what's more, our code only supports capitalized phone labels. However, users can modify the code to support other phone sets (see section 3.2).

1.2 Vowel and Rhyme

We extract the vowel and the last rhyme from the phone alignment file and store them in separate *TextGrid* files, i.e., each vowel or rhyme is considered as a interval with a label of 'vowel' or 'rhyme' and a starting and an ending time. The other intervals are unlabeled (i.e., they are blank). Currently our vowel phone set consists of AA, AE, AH, AO, AW, AX, AXR, AY, EH, ER, EY, IH, IX, IY, OW, OY, UH, UW, and changes can be made by modifying the code (in routine *isVowel* in script *code/routine.praat*, see section 3.2). We take the last rhyme to be the sequence of phones starting from the last vowel and covering all the remaining phones in the word.

1.3 VUV, Raw and Stylized Pitch, and Pitch Slope

Praat's existing pitch tracking and stylization functionality is one of the reasons for us to choose to build the tool based on *Praat*. Additionally, it provides various native objects and operations for holding and accessing pitch related information. Below is a listing of how pitch information is represented in *Praat* data structures.

- Raw Pitch (*PitchTier*): We rely on *Praat*'s autocorrelation based pitch tracking algorithm to extract raw pitch values, using gender-dependent pitch range (75-300 Hz for male, and 100-600 Hz for female). This is simply accomplished by using the command "To Pitch (ac)..." on the sound object to obtain a *Pitch* object. Pitch values are further smoothed by the command "Smooth...", and stored in a *PitchTier* by using the command "Down to PitchTier". *Praat* provides several useful functions for operating on the *PitchTier*, which makes it a simple matter to access pitch values for each frame.
- VUV (*TextGrid*): VUV is the voiced and unvoiced region segmentation. It is obtained by first using the command "To PointProcess" on the *Pitch* object, and then using the command "To TextGrid (vuv)... 0.02 0.01" on the newly generated PointProcess object.
- Stylized Pitch (*PitchTier*): *Praat*'s pitch stylization function ("Stylize... 4.0 Semitones") is used to stylize raw F_0 values over each voiced region. After stylization, only the slope changing

[1]Researchers can choose from a variety of alignment systems, such as Aligner [14], ISIP ASR [13], and SONIC [10].

points of the pitch contour remain in the PitchTier. Interpolated pitch values between each pair of changing points are inserted back to form the stylized pitch contour.

- Pitch Slope (*TextGrid*): We use an interval *TextGrid* to store slope values. Each non-empty interval covers successive frames with the same pitch slope based on stylization, and is labeled with the slope value.

1.4 Raw and Stylized Energy, and the Energy Slope

Praat has no built-in functionality for energy stylization. To simplify our implementation, we represent the energy values in *PitchTier* format so that we are able to use *Praat*'s stylization function to stylize energy values, and the routines for extracting F_0 features are then reused for extracting energy features. The raw energy is obtained by using the command "To Intensity..." on the sound object, then the tool creates a blank *PitchTier* object and inserts the energy values into the newly created *PitchTier* one frame at a time. Note that some intensity values (in dB) may be negative, which are illegal pitch values. To address this, we reset all intensity values lower than 1 to 1 to prevent negative pitch value[2]. After this transformation, we process energy similarly to pitch, except that stylization is applied directly to the entire tier rather than separately to segments since there is no VUV counterpart in the energy case. The stylization command is now "Stylize... 3.0 Hz" since energy has a smaller dynamic range.

1.5 Statistics

The statistics used in the model include the means and variances of phone length, vowel length, rhyme length, pause length, and speaker features related to pitch and energy. There are two different types of statistics, i.e., global and local statistics, which are differentiated by the scope of data over which the statistics are computed. See Appendix A.3 for more information about the statistics.

The global statistics relate to all sessions, either speaker dependent or independent across all speakers. The speaker specific phone duration statistics, and the pitch and energy related statistics are computed for each specific speaker across all the sessions of the speaker, and the global phone duration statistics are computed across all sessions and all speakers. Since global statistics need to be accumulated across sessions, we compute them before the feature extraction step.

The local statistics are session dependent statistics, which are computed during the features extraction process. They include the means and variances of the last rhyme duration, the last rhyme phone duration, the normalized last rhyme duration, and the pause duration.

[2]We assume that 1 dB is small enough to replace intensity values lower than 1 dB.

Chapter 2

Using the Tool

In our code design scheme, *stats_batch.praat* and *main_batch.praat* are the two scripts that accept configuration inputs respectively for global statistics computation and prosodic feature extraction. Due to the inherent functionality of *Praat*, they both can be launched from the command line (of *nix) which is most suitable for batch processing or can be run in graphic mode through the *Praat ScriptEditor*. In this chapter we focus on the usage of this tool, and give instructions step by step.

2.1 Global Statistics Computation

There are several statistics that need to be computed in advance before prosodic feature extraction to enable the normalization of related prosodic features. The mean and variance of the phone duration across the whole data set are examples of these statistics. In addition, statistics with respect to each speaker (e.g., the mean and variance of the phone duration of each speaker across sessions and the statistics related to each speaker's pitch and energy information.) are also gathered in this step. Please refer to Appendix A.3 for more information. Note that the statistics on each session are computed along with the feature extraction process described in the next section.

A metadata file is needed to provide the session ID, speaker ID, gender, and the path (which can be absolute path or relative path relative to the *Praat* script) to the audio file. Our tool supports multiple sessions per speaker and makes use of the speaker information across the sessions for normalization. The *TextGrid* format word/phone alignments are assumed to be in the same directory as the audio file, and their file names (as well as names for the other files generated for that audio recording) are hard-coded based on the name of the audio file. For example, if the audio file is *../demo/data/demo_C.wav*[1], then the word and phone alignment files are both located at directory *../demo/data*, and are named *demo_C-word.TextGrid* and *demo_C-phone.TextGrid* respectively. Below is an example metadata file *demo-wavinfo_list.txt*[2] :

[1]The path is relative to the *Praat* script. This package comes along with a demo on running the tool. The default path settings in the *Praat* scripts and the metadata file *demo-wavinfo_list.txt* are configured to run the demo on any *nix platform. Slight change of the path delimiter from "/" to "\" is needed if running the demo on a Windows OS.

[2]The headings (i.e., SESSION, SPEAKER, GENDER, and WAVEFORM) are needed.

SESSION	SPEAKER	GENDER	WAVEFORM
demo_C	C	female	../demo/data/demo_C.wav
demo_D	D	male	../demo/data/demo_D.wav
demo_E	E	male	../demo/data/demo_E.wav
demo_F	F	male	../demo/data/demo_F.wav
demo_G	G	male	../demo/data/demo_G.wav

The main script *stats_batch.praat* for computing statistics has the following arguments:

- **audio info table**: path to the metadata file described above.
- **working directory**: the directory for storing parameter files (under a subdirectory *param_files*) and statistics files (under a subdirectory *stats_files*). If the statistics directory already exists, then it is cleaned up. These parameter and statistics files are created during the process of statistics computation.
- **using existing param files**: choose "yes" or "no". If this option is set to "yes", the tool attempts to verify the existence of the parameter files, and uses these existing files in the parameter directory or generates the parameter files on the fly if they don't exist. If this option is set to "no", the tool always generates parameter files no matter whether they exist or not.

Here is an example of running *stats_batch.praat* at the command line:

praat stats_batch.praat ../demo-wavinfo_list.txt ../demo/work_dir yes

Below are the steps to run the same example in the *Praat ScriptEditor*:

1. Run *Praat*.
2. Open *stats_batch.praat* from "Read→Read from file..." on the menu of *Praat Objects*.
3. Click "Run→Run" on the menu of *ScriptEditor*.
4. Enter parameters. Type *../demo-wavinfo_list.txt* and *../demo/work_dir* in the two boxes, and then check "yes" in the radio box.
5. Click "OK" to start processing with the configurations or "Cancel" to close the interface. Clicking the "Apply" button (if available) also starts processing but it keeps the interface on after the work is done. The "Standards" button (if available) gives the option to restore the default configurations. Please refer to the *Praat* manual [2] for details.
6. Process related information is displayed in the *Praat Info* Window.

After computation is complete, the statistics files can be found at *../demo/work_dir/stats_files*.

2.2 Prosodic Feature Extraction

Once the global statistics are computed the tool can proceed to compute the prosodic features. Although our tool is able to produce all the features described in Appendix A, there is an option to limit the output to a selected set of prosodic features. Below are several pre-defined feature classes[3]:

[3]Each class is defined in a seperate file under *code/pf_list_files*. Currently, all the features are computed in the tool even though a subset of features are selected for output. One may also choose to output all the features, and select the desired features by other means.

FULL_FEATURE
BASIC_FEATURE
BASIC_BASE_FEATURE
BASIC_DUR_FEATURE
BASIC_F0_FEATURE
BASIC_ENERGY_FEATURE
DERIVE_FEATURE
DERIVE_NORMALIZED_WORD
DERIVE_NORMALIZED_PAUSE
DERIVE_NORMALIZED_VOWEL
DERIVE_NORMALIZED_RHYME
DERIVE_F0_FEATURE
DERIVE_ENERGY_FEATURE
DERIVE_AVERAGE_PHONE

The desired output features can be selected by including them in the "output prosodic feature selection list" file. This file is a one-column table with "FEATURE_NAME" as the column label in the first line and followed by one feature name or one feature class name per line. It is a convention that the name for a string feature, such as 'GEN$', ends with symbol '$', and the name for a numeric feature does not have a '$' in the end. Below is an example of the "output prosodic feature selection list" file[4].

FEATURE_NAME
WORD$
WAV$
SPKR_ID$
GEN$
PAUSE_DUR
NORM_LAST_RHYME_DUR
DERIVE_FEATURE

The main script for computing prosodic features is *main_batch.praat*, which has the following arguments:

- **audio info table**: this is the same metadata file used in *stats_batch.praat*. It contains session ID, speaker ID, gender, and the path to the audio file.
- **output prosodic feature selection list**: the list file described above.
- **statistics directory**: the directory containing the statistics files produced by *stats_batch.praat*.
- **working directory (storing files)**: the directory for storing parameter files (under subdirectory *param_files*), local statistics files (under subdirectory *stats_files*), and prosodic feature files (under subdirectory *pf_files*).
- **use existing param files**: choose "yes" or "no", similar to the option in *stats_batch.praat*.

Here is an example of running *main_batch.praat* at the command line:

[4]In this file, FEATURE_NAME is the required column label; WORD$, WAV$, SPKR_ID$, GEN$, PAUSE_DUR, and NORM_LAST_RHYME_DUR are features defined in Appendix A; DERIVE_FEATURE is a feature class that tells the tool to output all the dereived prosodic features.

*praat main_batch.praat ../demo-wavinfo_list.txt user_pf_name_table.Tab *
../demo/work_dir/stats_files ../demo/work_dir yes

Below are the steps to run the same example in the *Praat ScriptEditor*:

1. Run *Praat*.
2. Open *main_batch.praat* from "Read→Read from file..." on the menu of *Praat Objects*.
3. Click "Run→Run" on the menu of *ScriptEditor*.
4. Enter parameters. Type *../demo-wavinfo_list.txt*, *user_pf_name_table.Tab*, *../demo/work_dir/stats_files*, and *../demo/work_dir* one by one in the four boxes, and then check "yes" in the radio box.
5. Click "OK" to start processing with the configurations or "Cancel" to close the interface. Clicking the "Apply" button (if available) also starts processing but it keeps the interface on after the work is done. The "Standards" button (if available) gives the option to restore the default configurations. Please refer to the *Praat* manual [2] for details.
6. Process related information is displayed in the *Praat Info* Window.

After computation is complete, the prosodic feature files can be found at *../demo/work_dir/pf_files*.

Chapter 3

Architecture of the Tool and Its Potential Augmentation

Our initial objective for implementing this tool is not only to utilize many of the features that have been used in other research efforts to support our research but also to make use of the flexibility, popularity, and extensibility of *Praat* to incorporate other useful features which can be computed with or by augmenting the current or future versions of *Praat*. In this chapter, we present the current structure of our tool and the organization of the code, and provide an example of adding new features.

3.1 Structure

As discussed in the previous chapters and illustrated in Figure 3.1, the procedures of our tool include:

- **Global Statistics Computation**: This module should be run prior to the feature extraction process in order to provide the global statistics needed for normalization. Although it is not illustrated in Figure 3.1, it also contains a pre-processing phase in which the basic elements, i.e., the parameter files, are extracted based on the audio files and the alignments. These basic elements can be reused later by activating the "using existing files" option when configuring *main_batch.praat*.
- **Feature Extraction**: After obtaining the global statistics, the tool proceeds to extract the prosodic features by following the procedures below:
  - **Tool Initialization**
  - **Performing the following steps for each audio file**:
    - **Initialization**
    - **Pre-processing**
    - **Basic Feature Extraction**
    - **Local Statistics Calculation**
    - **Derived Feature Computation**
    - **Clean-up**
  - **Tool Clean-up and Termination**

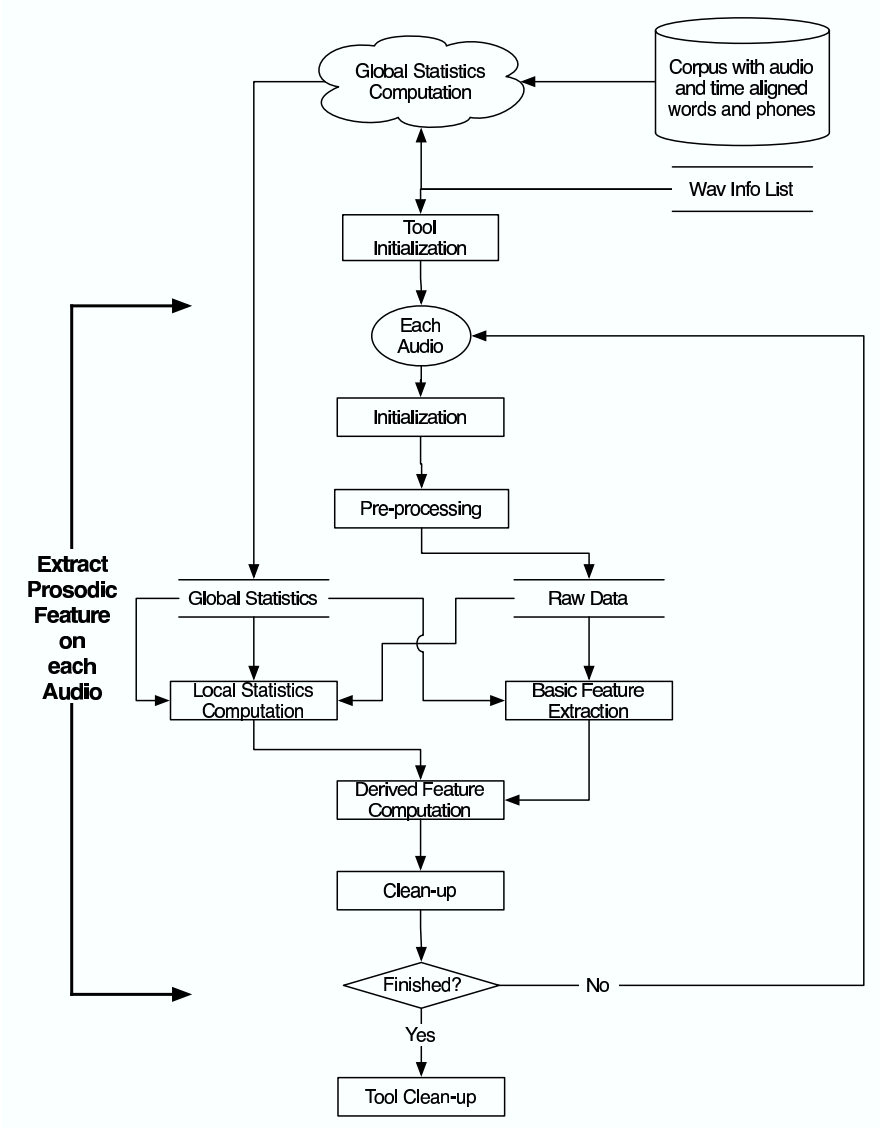

Figure 3.1: Data flow diagram for the tool.

3.2 Code Organization

The *Praat* code consists several scripts, each focusing on a certain type of processing. We list each script below and for each give a brief description about the included routines. For simplicity, we separate the scripts for computing the statistics from those used for feature extraction into different directories.

- **Scripts for Computing Global Statistics**:
  - stats/stats_batch.praat: interface, accepts inputs, and controls the overall operation.
  - stats/operations.praat: the highest level of operation flow.
  - stats/io.praat: contains routines for controlling file input/output in *Praat*.
  - stats/table.praat: contains routines for controlling *Praat Table* operations. We use *Praat Table*s for holding various intermediate values, and have designed various operations, such as table creation, value updating, value retrieval, etc., to handle these values.
  - stats/stats.praat: contains routines for computing statistics.
  - stats/routine.praat: contains routines for obtaining various basic elements.
  - stats/utils.praat: contains some miscellaneous utility routines.
  - stats/config.praat: contains the configuration of the pre-defined parameter values, such as frame and window size, default file names, etc.
- **Scripts for Extracting Prosodic Features**:
  - code/main_batch.praat: interface, accepts inputs, and controls the overall operation.
  - code/operations.praat: the highest level of operation flow.
  - code/io.praat: contains routines for controlling file input/output in *Praat*.
  - code/table.praat: contains routines for controlling *Praat Table* operations. We use *Praat Table*s for holding various intermediate values, and have designed various operations, such as table creation, value updating, value retrieval, etc., to handle these values.
  - code/fetch.praat: contains higher level routines for extracting basic prosodic features, by calling routines in routine.praat.
  - code/routine.praat: contains routines for obtaining various basic elements, and lower level routines that implement feature extraction and support the higher level routines in fetch.praat.
  - code/derive.praat: contains routines for computing derived features.
  - code/utils.praat: contains some miscellaneous utility routines.
  - code/config.praat: contains the configuration of the pre-defined parameter values, such as frame and window size, default file names, etc.
  - code/pf_list_files/feature_name_table.Tab: contains a list of feature names which are implemented in our tool. The other files in the same directory as this file contain lists of feature names for different type of features (e.g., basic F_0 features, derived F_0 features).

While the above brief description reveals some of the main functionality that each script can perform, the boundaries between them are somewhat vague. Some of the routines could have been put into one script or another, and there are also some scripts that contain extra operations to simplify our coding effort. We will try to make the organization of code more clear in future release so that it would be easier for users to modify the code.

3.3 An Augmentation Example

As we said above, we would like to make this tool easily extensible. An example is best to show this. Currently, *Praat* doesn't have built-in functions to build a Fujisaki model, which decomposes the pitch contour into phrase contours and accent contours. It is believed that features based on these superpositional contours should be helpful for a variety of prosodic analysis tasks. If the Fujisaki analysis became available in *Praat*, we could use the following steps to implement new features capitalizing on this new functionality. For simplicity, we assume that no statistics are needed for the new features.

1. Define the feature names, and list (append) them at *code/pf_list_files/feature_name_table.Tab*.
2. Write code in *code/routine.praat* to perform Fujisaki analysis on the audio, just like we did for stylization. The phrase and accent contours can be stored in two *Praat PitchTier* objects for later access.
3. Write code in *code/operations.praat* and *code/io.praat* to ensure that the Fujisaki analysis is performed appropriately in the Pre-processing step.
4. Write code in *code/fetch.praat* and *code/routine.praat* to implement details of extracting features based on the phrase and accent contours.
5. Write code in *code/derive.praat* to compute the derived features based on the basic features on the contours.

Appendix A

Prosodic Feature List

A.1 Introduction

In our tool use scenario, we have a set of audio files, and each audio file has its word and phone alignments. There is exactly one speaker in each audio, although the same speaker can appear in several audio files. Since most of our current research focuses on sentence boundary detection, the prosody features are extracted around each word boundary. Here are some definitions that will be used throughout for describing features.

- Frame: In *Praat* pitch and energy are calculated on each frame, the length of which is set to 0.01s by default. The start/end time and the duration of an object are measured by the index or the number of frames in the waveform.
- Boundary: Prosodic features are calculated around each boundary, which is the end of a word. Feature extraction is based on the preceding and following words, and the preceding and following windows that has a size of N frames.
- Window: Some features are computed within a window preceding or following a boundary. The window size N is set to 0.2s by default. If there are not enough frames in the beginning or at the end of a waveform to make a full size window, then the maximum size window is used.
- Missing value: There are situations where some features are not available, e.g., the maximum stylized pitch value is not available for an unvoiced region. When it happens, a "?" is used to denote that missing value.

A.2 Basic Features

The features described here have been inspired by [3], however, our implementation may differ. Each feature is computed in terms of a boundary in the waveform under consideration.

A.2.1 Base Features

- WAV: The path to the corresponding (current) audio file.

- SPK_ID: The speaker identification label for the current waveform.
- SPK_GEN: The gender of the speaker.

A.2.2 Duration Features

- WORD: The word preceding a boundary.
- WORD_START: The start time (hereafter, start/end time is measured by the index of frame) of the word preceding a boundary.
- WORD_END: The end time of the word preceding a boundary.
- FWORD: The word following a boundary.
- FWORD_START: The start time of the word following a boundary.
- FWORD_END: The end time of the word following a boundary.
- PAUSE_START: The start time of the pause around a boundary. Its value is set to the end time of the preceding word since the boundary is defined at the end of the preceding word.
- PAUSE_END: The end time of the pause around a boundary. Its value is set to the start time of the following word. If there is no following word, i.e., it appears at the end of the waveform, then it is set to the end time of the waveform.
- PAUSE_DUR: The duration of the pause around a boundary. PAUSE_DUR = PAUSE_END - PAUSE_START.
- WORD_PHONES: The phones in the WORD, with their durations (hereafter, duration is measured by the number of frames). The format is:

  phone1:duration1_phone2:duration2_...

- FLAG: This feature indicates whether the word before a boundary has reliable phone durations. If the duration of any of the phones in that word is larger than a specific threshold (obtained from the *phone_dur.stat* file) for that phone, then this feature is set to "SUSP" (suspicious word); if the threshold for any of the phones in the word is missing or the word does not contain phones, the value is set to "?"; otherwise, it is set to 0.
- LAST_VOWEL: The last vowel in the word preceding a boundary. If it doesn't exist, then all the related features are set to "?". This is the default treatment for the features whose values are not available.
- LAST_VOWEL_START: The start time of the last vowel in the word preceding a boundary.
- LAST_VOWEL_END: The end time of the last vowel in the word preceding a boundary.
- LAST_VOWEL_DUR: The duration of the last vowel in the word preceding a boundary.
- LAST_RHYME_START: The start time of the last rhyme in the word preceding a boundary. The last rhyme is considered as the sequence of phones starting with the last vowel to the end of the word.
- LAST_RHYME_END: The end time of the last rhyme in the word preceding a boundary. It is the same as WORD_END.

- NORM_LAST_RHYME_DUR = $\sum_{every\_phone\_in\_word} \frac{dur(phone)-mean(phone)}{std\_dev(phone)}$ where $dur(phone)$ is the duration of the phone in the current audio and $mean(phone)$ and $std\_dev(phone)$ are the average duration of the phone and the standard deviation of the duration of that phone in the training data (both values are obtained from the *phone_dur.stats* file).
- PHONES_IN_LAST_RHYME: The total number of phones in the last rhyme.

A.2.3 F_0 Features

Features computed using the raw F_0 extracted by *Praat*:

- MIN_F0: The minimum raw F_0 value for the word preceding a boundary.
- MAX_F0: The maximum raw F_0 value for the word preceding a boundary.
- MEAN_F0: The mean raw F_0 value for the word preceding a boundary.
- The following features are the same as the corresponding features without "_NEXT" except these are computed for the word after a boundary.
  - MIN_F0_NEXT
  - MAX_F0_NEXT
  - MEAN_F0_NEXT
- The following features are the same as the corresponding features without "_WIN" except in these cases, the values are computed over the N frames before a boundary. Maximum number of frames are used if there is not enough data.
  - MIN_F0_WIN
  - MAX_F0_WIN
  - MEAN_F0_WIN
  - MIN_F0_NEXT_WIN
  - MAX_F0_NEXT_WIN
  - MEAN_F0_NEXT_WIN

Features computed using stylized F_0:

- MIN_STYLFIT_F0: The minimum stylized F_0 value for the word preceding a boundary.
- MAX_STYLFIT_F0: The maximum stylized F_0 value for the word preceding a boundary.
- MEAN_STYLFIT_F0: The mean stylized F_0 value for the word preceding a boundary.
- FIRST_STYLFIT_F0: The first stylized F_0 value for the word preceding a boundary.
- LAST_STYLFIT_F0: The last stylized F_0 value for the word preceding a boundary.
- The following features are the same as the corresponding features without "_NEXT" except these are computed for the word after a boundary.
  - MIN_STYLFIT_F0_NEXT

  - MAX_STYLFIT_F0_NEXT
  - MEAN_STYLFIT_F0_NEXT
  - FIRST_STYLFIT_F0_NEXT
  - LAST_STYLFIT_F0_NEXT

- The following features are the same as the corresponding features without "_WIN" except in these cases, the values are computed over the N frames before or after a boundary. Maximum number of frames are used if there is not enough data.

  - MIN_STYLFIT_F0_WIN
  - MAX_STYLFIT_F0_WIN
  - MEAN_STYLFIT_F0_WIN
  - FIRST_STYLFIT_F0_WIN
  - LAST_STYLFIT_F0_WIN
  - MIN_STYLFIT_F0_NEXT_WIN
  - MAX_STYLFIT_F0_NEXT_WIN
  - MEAN_STYLFIT_F0_NEXT_WIN
  - FIRST_STYLFIT_F0_NEXT_WIN
  - LAST_STYLFIT_F0_NEXT_WIN

Stylized F_0 contour slope features:

- PATTERN_WORD: This feature is composed of a sequence of "f", "u", and "r" representing a *falling slope*, an *unvoiced section* and a *rising slope* in the word preceding a boundary. Any slope or unvoiced section that contains less than *min_frame_length* frames is skipped. Note that sequences of f's (or r's) with different slopes are represented as ff (or rr).

- PATTERN_WORD_CALLAPSED: Similar to PATTERN_WORD, except that consecutive f's (or r's) are combined into one f (or r).

- PATTERN_SLOPE: Similar to PATTERN_WORD, but instead of the sequence of f's (or r's), a sequence of slope values are listed.

- The following features are the same as the corresponding features without "_NEXT" except these are computed for the word after a boundary.

  - PATTERN_WORD_NEXT
  - PATTERN_WORD_COLLAPSED_NEXT
  - PATTERN_SLOPE_NEXT

- The following features are the same as the corresponding features without "_WIN" except in these cases, the values are computed over the N frames before or after a boundary. Maximum number of frames are used if there is not enough data.

  - PATTERN_WORD_WIN
  - PATTERN_WORD_COLLAPSED_WIN
  - PATTERN_SLOPE_WIN
  - PATTERN_WORD_NEXT_WIN

  - PATTERN_WORD_CALLAPSED_NEXT_WIN
  - PATTERN_SLOPE_NEXT_WIN

There are also several features that involve counting:

- NO_PREVIOUS_SSF: Number of previous consecutive frames inside the word which have the same slope as last voiced frame in the word before a boundary (voiced sequences of less than min_frame_length are not considered).
- NO_PREVIOUS_VF: Number of consecutive "voiced" frames inside the word from the last voiced frame in the word backwards (voiced sequences of less than min_frame_length are not considered).
- NO_FRAMES_LS_WE: Number of consecutive frames between the last voiced frame which belongs to a sequence of voiced frames larger than min_frame_length in the word preceding a boundary and the end of that word.
- NO_SUCCESSOR_SSF: Number of successor consecutive frames inside the word which have the same slope as the first voiced frame in the word preceding a boundary (voiced sequences of less than min_frame_length are not considered).
- NO_SUCCESSOR_VF: Number of consecutive "voiced" frames inside the word from the first voiced frame in the word forward (voiced sequences of less than min_frame_length are not considered).
- NO_FRAMES_WS_FS: Number of consecutive frames between the first frame of the word preceding a boundary and the first voiced frame in that word which belongs to a sequence of voiced frames larger than min_frame_length.
- The following features are the same as the corresponding features without "_NEXT" except these are computed for the word after a boundary.
  - NO_PREVIOUS_SSF_NEXT
  - NO_PREVIOUS_VF_NEXT
  - NO_FRAMES_LS_WE_NEXT
  - NO_SUCCESSOR_SSF_NEXT
  - NO_SUCCESSOR_VF_NEXT
  - NO_FRAMES_WS_FS_NEXT
- The following features are the same as the corresponding features without "_WIN" except in these cases, the values are computed over the N frames before or after a boundary. Maximum number of frames are used if there is not enough data.
  - NO_PREVIOUS_SSF_WIN
  - NO_PREVIOUS_VF_WIN
  - NO_FRAMES_LS_WE_WIN
  - NO_SUCCESSOR_SSF_WIN
  - NO_SUCCESSOR_VF_WIN
  - NO_FRAMES_WS_FS_WIN

- NO_PREVIOUS_SSF_NEXT_WIN
- NO_PREVIOUS_VF_NEXT_WIN
- NO_FRAMES_LS_WE_NEXT_WIN
- NO_SUCCESSOR_SSF_NEXT_WIN
- NO_SUCCESSOR_VF_NEXT_WIN
- NO_FRAMES_WS_FS_NEXT_WIN

Features extracted concerning word boundaries:

- PATTERN_BOUNDARY: The last *f*, *r*, or *u* in the PATTERN_WORD concatenated with the first *f*, *r*, or *u* in the PATTERN_NEXT_WORD.
- SLOPE_DIFF: The difference between the last non-zero (longer than *min_frame_length*) slope of the word and the first non-zero (longer than *min_frame_length*) slope of the next word. If one of the words does not have a non-zero slope that occurs over more than *min_frame_length* frames, then this feature receives a value "?". Note again "?" is the default value for these unavailable features.

A.2.4 Energy Features

The basic energy features are computed similarily as the basic F_0 features. Below is the list of the basic energy features.

- MIN_ENERGY
- MAX_ENERGY
- MEAN_ENERGY
- MIN_ENERGY_NEXT
- MAX_ENERGY_NEXT
- MEAN_ENERGY_NEXT
- MIN_ENERGY_WIN
- MAX_ENERGY_WIN
- MEAN_ENERGY_WIN
- MIN_ENERGY_NEXT_WIN
- MAX_ENERGY_NEXT_WIN
- MEAN_ENERGY_NEXT_WIN
- MIN_STYLFIT_ENERGY
- MAX_STYLFIT_ENERGY
- MEAN_STYLFIT_ENERGY

- FIRST_STYLFIT_ENERGY
- LAST_STYLFIT_ENERGY
- MIN_STYLFIT_ENERGY_NEXT
- MAX_STYLFIT_ENERGY_NEXT
- MEAN_STYLFIT_ENERGY_NEXT
- FIRST_STYLFIT_ENERGY_NEXT
- LAST_STYLFIT_ENERGY_NEXT
- MIN_STYLFIT_ENERGY_WIN
- MAX_STYLFIT_ENERGY_WIN
- MEAN_STYLFIT_ENERGY_WIN
- FIRST_STYLFIT_ENERGY_WIN
- LAST_STYLFIT_ENERGY_WIN
- MIN_STYLFIT_ENERGY_NEXT_WIN
- MAX_STYLFIT_ENERGY_NEXT_WIN
- MEAN_STYLFIT_ENERGY_NEXT_WIN
- FIRST_STYLFIT_ENERGY_NEXT_WIN
- LAST_STYLFIT_ENERGY_NEXT_WIN
- ENERGY_PATTERN_WORD
- ENERGY_PATTERN_WORD_CALLAPSED
- ENERGY_PATTERN_SLOPE
- ENERGY_PATTERN_WORD_NEXT
- ENERGY_PATTERN_WORD_CALLAPSED_NEXT
- ENERGY_PATTERN_SLOPE_NEXT
- ENERGY_PATTERN_WORD_WIN
- ENERGY_PATTERN_WORD_CALLAPSED_WIN
- ENERGY_PATTERN_SLOPE_WIN
- ENERGY_PATTERN_WORD_NEXT_WIN
- ENERGY_PATTERN_WORD_CALLAPSED_NEXT_WIN
- ENERGY_PATTERN_SLOPE_NEXT_WIN
- ENERGY_PATTERN_BOUNDARY
- ENERGY_SLOPE_DIFF

A.3 Statistical Tables

- phone_dur.stats: For each phone, the table contains the mean phone duration, the standard deviation of the phone duration, the number of occurrences of that phone in the training database, and the phone duration threshold computed as follows:
  $threshold(phone) = mean(phone) + 10 * std\_dev(phone)$

- pause_dur.stats: For each audio, the listed features are the mean and standard deviation of the pauses in the training database.

- spkr_feat.stats: This table has one row for each speaker. Each row contains a variety of statistics related to consecutive voiced and unvoiced frames, F_0 and F_0 slope, energy and energy slope. Note that these F_0 and energy values are in logarithm (base e). These are described below:

  - MEAN_VOICED: The average length of the voiced sections inside the uttered words for all of the audio corresponding to the speaker (for sequences of voiced frames longer than *min_frame_length*).
  - STDEV_VOICED: The standard deviation of the voiced sections inside the uttered words for all of the audio corresponding to the speaker (for sequences of voiced frames longer than *min_frame_length*).
  - COUNT_VOICED: The number of voiced sections inside the uttered words for all audio corresponding to the speaker (for sequences of voiced frames longer than *min_frame_length*).
  - MEAN_UNVOICED: The average length of the unvoiced sections inside the uttered words for all of the audio corresponding to the speaker (for sequences of unvoiced frames longer than *min_frame_length*).
  - STDEV_UNVOICED: The standard deviation of the unvoiced sections inside the uttered words for all of the audio corresponding to the speaker (for sequences of unvoiced frames longer than *min_frame_length*).
  - COUNT_UNVOICED: The number of unvoiced sections inside the uttered words for all of the audio corresponding to the speaker (for sequences of unvoiced frames longer than *min_frame_length*).
  - MEAN_PITCH: The average F_0 value over the uttered words in all of the audio corresponding to the speaker.
  - STDEV_PITCH: The standard deviation F_0 value over the uttered words in all of the audio corresponding to the speaker.
  - COUNT_PITCH: the number of F_0 value counted over the uttered words in all of the audio corresponding to the speaker.
  - MEAN_SLOPE: The mean pitch slope over the uttered words in all of the audio corresponding to the speaker. It is computed only over the sequences of frames that have the same slope for more than *min_frame_length frames*.
  - STDEV_SLOPE: The standard deviation of the pitch slope over the uttered words in all of the audio corresponding to the speaker. It is computed only over the sequences of frames that have the same slope for more than *min_frame_length frames*.
  - COUNT_SLOPE: The number of pitch slope counted over the uttered words in all of the audio corresponding to the speaker. It is computed only over the sequences of frames that have the same slope for more than *min_frame_length frames*.

  - MEAN_ENERGY: The average energy value over the uttered words in all of the audio corresponding to the speaker.
  - STDEV_ENERGY: The standard deviation F_0 value over the uttered words in all of the audio corresponding to the speaker.
  - COUNT_ENERGY: the number of F_0 value counted over the uttered words in all of the audio corresponding to the speaker.
  - MEAN_ENERGY_SLOPE: The mean slope over the uttered words in all of the audio corresponding to the speaker. It is computed only over the sequences of frames that have the same slope for more than *min_frame_length frames*.
  - STDEV_ENERGY_SLOPE: The standard deviation of the slope over the uttered words in all of the audio corresponding to SPKR. It is computed only over the sequences of frames that have the same slope for more than *min_frame_length frames*.
  - COUNT_ENERGY_SLOPE: The number of energy slope counted over the uttered words in all audio corresponding to the speaker. It is computed only over the sequences of frames that have the same slope for more than *min_frame_length frames*.

- spkr_phone_dur.stats: One table for each speaker. These tables are similar to *phone_dur.stats*, but they involve all of the audio corresponding to the speaker.

- last_rhyme_dur.stats: For each audio, the listed features are the mean duration of the last rhyme, the standard deviation of the last rhyme duration, and the number of last rhymes used in the computation of these statistics.

- last_rhyme_phone_dur.stats: For each audio, the listed features are mean phone duration for the phones in the last rhyme, the standard deviation of the phone duration for the phones in the last rhyme, and the number of last rhymes used in the computation for these statistics.

- pause_dur.stats: This table has a row for each audio. The first feature is the speaker session id. The other features in the table are:
  - MEAN: The mean duration of the pauses in the audio.
  - STDEV: The standard deviation of the duration of the pauses in the audio.
  - MEAN_LOG: The mean of the (base e) log pause duration in the audio.
  - STDEV_LOG: The standard deviation of the log pause duration in the audio.
  - COUNT_PAUSE: The number of pauses in the audio.

A.4 Derived Features

Derived features are computed from the previously described basic features and statistics. Some derived features are computed given two basic features, such as log difference or log ratio of two values. Some derived features are normalized basic features using the computed means and standard deviations.

A.4.1 Normalized Word Duration

- WORD_DUR = WORD_END − WORD_START, where WORD_END and WORD_START are basic features.

- WORD_AV_DUR = $\sum_{every\_phone\_in\_word} mean(phone)$
  where $mean(phone)$ is obtained from the statistical table *phone_dur.stats* and the phones are from the basic features WORD_PHONES.
- NORM_WORD_DUR = WORD_DUR / WORD_AV_DUR

A.4.2 Normalized Pause

- PAU_DUR_N = PAU_DUR / PAUSE_MEAN where PAUSE_MEAN comes from the *pause_dur.stats* (column MEAN) from the line corresponding to the current audio.

A.4.3 Normalized Vowel Duration

- LAST_VOWEL_DUR_Z =
  (LAST_VOWEL_DUR - ALL_PHONE_DUR_MEAN) / ALL_PHONE_DUR_STDEV
- LAST_VOWEL_DUR_N = LAST_VOWEL_DUR / ALL_PHONE_DUR_MEAN
- LAST_VOWEL_DUR_ZSP =
  (LAST_VOWEL_DUR - SPKR_PHONE_DUR_MEAN) / SPKR_PHONE_DUR_STDEV
- LAST_VOWEL_DUR_NSP = LAST_VOW_DUR/ SPKR_PHONE_DUR_MEAN

Where:

- LAST_VOWEL_DUR is a basic duration feature,
- ALL_PHONE_DUR_MEAN and ALL_PHONE_DUR_STDEV are statistics taken from table *phone_dur.stats* for the line corresponding to LAST_VOWEL (another basic feature),
- SPKR_PHONE_DUR_MEAN and SPKR_PHONE_DUR_STDEV are statistics taken from table *SPKR_ID–phone_dur.stats* (where SPKR_ID is a base feature) for the line corresponding to LAST_VOWEL.

A.4.4 Normalized Rhyme Duration

- LAST_RHYME_DUR_PH = LAST_RHYME_DUR / PHONES_IN_LAST_RHYME
- LAST_RHYME_DUR_PH_ND =
  (LAST_RHYME_DUR / PHONES_IN_LAST_RHYME) - LAST_RHYME_PHONE_DUR_MEAN
- LAST_RHYME_DUR_PH_NR =
  (LAST_RHYME_DUR / PHONES_IN_LAST_RHYME) / LAST_RHYME_PHONE_DUR_MEAN
- LAST_RHYME_NORM_DUR_PH = NORM_LAST_RHYME_DUR / PHONES_IN_LAST_RHYME
- LAST_RHYME_NORM_DUR_PH_ND =
  (NORM_LAST_RHYME_DUR / PHONES_IN_LAST_RHYME) -
  NORM_LAST_RHYME_PHONE_DUR_MEAN
- LAST_RHYME_NORM_DUR_PH_NR =
  (NORM_LAST_RHYME_DUR / PHONES_IN_LAST_RHYME) /
  NORM_LAST_RHYME_PHONE_DUR_MEAN

- LAST_RHYME_DUR_WHOLE_ND = LAST_RHYME_DUR - LAST_RHYME_WHOLE_DUR_MEAN
- LAST_RHYME_WHOLE_DUR_NR = LAST_RHYME_DUR / LAST_RHYME_WHOLE_DUR_MEAN
- LAST_RHYME_WHOLE_DUR_Z = (LAST_RHYME_DUR - LAST_RHYME_WHOLE_DUR_MEAN) / LAST_RHYME_WHOLE_DUR_STDEV

where:

- LAST_RHYME_DUR, PHONES_IN_LAST_RHYME, and NORM_LAST_PHYME_DUR are duration features,
- LAST_RHYME_PHONE_DUR_MEAN is taken from table *last_rhyme_phone_dur.stats* (column MEAN) for the line corresponding to the audio,
- NORM_LAST_RHYME_PHONE_DUR_MEAN is taken from table *norm_last_rhyme_phone_dur.stats* (column MEAN) for the line corresponding to the audio.
- LAST_RHYME_WHOLE_DUR_MEAN and LAST_RHYME_WHOLE_DUR_STDEV are taken from table *last_rhyme_dur.stats* (column MEAN and STDEV) for the line corresponding to the audio.

A.4.5 F_0 Derived Features

- F_0 characteristics of the speaker: The SRI prosodic model uses a pitch model to estimate several values to characterize the speaker's pitch. Since our model is based on *Praat*'s built-in function for stylization, we do not have counterparts for some of the pitch characteristics provided by the SRI's model. However, in order to compute the derived features similar to these defined in SRI's model, we chose to approximate these characteristic values using the pitch statistics.
  - SPKR_FEAT_F0_MODE = exp(SPKR_F0_MEAN)
  - SPKR_FEAT_F0_TOPLN = .75 (exp(SPKR_F0_MEAN))
  - SPKR_FEAT_F0_BASELN = 1.5 (exp(SPKR_F0_MEAN))
  - SPKR_FEAT_F0_STDLN = exp(SPKR_F0_STDEV)
  - SPKR_FEAT_F0_RANGE = SPKR_FEAT_F0_TOPLN - SPKR_FEAT_F0_BASELN
- Log difference of the max, min, and mean stylized F_0 values, between the previous and the next word:
  - F0K_WORD_DIFF_HIHI_N = log (MAX_STYLFIT_F0 / MAX_STYLFIT_F0_NEXT)
  - F0K_WORD_DIFF_HILO_N = log (MAX_STYLFIT_F0 / MIN_STYLFIT_F0_NEXT)
  - F0K_WORD_DIFF_LOLO_N = log (MIN_STYLFIT_F0 / MIN_STYLFIT_F0_NEXT)
  - F0K_WORD_DIFF_LOHI_N = log (MIN_STYLFIT_F0 / MAX_STYLFIT_F0_NEXT)
  - F0K_WORD_DIFF_MNMN_N = log (MEAN_STYLFIT_F0 / MEAN_STYLFIT_F0_NEXT)

  where MAX_STYLFIT_F0, MAX_STYLFIT_F0_NEXT, MIN_STYLFIT_F0, MIN_STYLFIT_F0_NEXT, MEAN_STYLFIT_F0, MEAN_STYLFIT_F0_NEXT are all F_0 features.

- Log ratio of the maximum, minimum, and mean of the stylized F_0 values, between the previous and the next word, normalized by the pitch range:
  - F0K_WORD_DIFF_HIHI_NG = (log (MAX_STYLFIT_F0) / log (MAX_STYLFIT_F0_NEXT)) / SPKR_FEAT_F0_RANGE
  - F0K_WORD_DIFF_HILO_NG = (log (MAX_STYLFIT_F0) / log (MIN_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE
  - F0K_WORD_DIFF_LOLO_NG = (log (MIN_STYLFIT_F0) / log (MIN_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE
  - F0K_WORD_DIFF_LOHI_NG = (log (MIN_STYLFIT_F0) / log (MAX_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE
  - F0K_WORD_DIFF_MNMN_NG = (log (MEAN_STYLFIT_F0) / log (MEAN_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE
- Log difference of maximum, minimum, and mean of the stylized F_0 values, between the previous and the next window:
  - F0K_WIN_DIFF_HIHI_N = log (MAX_STYLFIT_F0_WIN / MAX_STYLFIT_F0_WIN_NEXT)
  - F0K_WIN_DIFF_HILO_N = log (MAX_STYLFIT_F0_WIN/ MIN_STYLFIT_F0_WIN_NEXT)
  - F0K_WIN_DIFF_LOLO_N = log (MIN_STYLFIT_F0_WIN / MIN_STYLFIT_F0_WIN_NEXT)
  - F0K_WIN_DIFF_LOHI_N = log (MIN_STYLFIT_F0_WIN / MAX_STYLFIT_F0_WIN_NEXT)
  - F0K_WIN_DIFF_MNMN_NG = log (MEAN_STYLFIT_F0_WIN / MEAN_STYLFIT_F0_WIN_NEXT)
- Log ratio of the maximum, minimum, and mean of the stylized F_0 values, between the previous and the next window, normalized by pitch range:
  - F0K_WIN_DIFF_HIHI_NG = (log (MAX_STYLFIT_F0_WIN) / log (MAX_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE
  - F0K_WIN_DIFF_HILO_NG = (log (MAX_STYLFIT_F0_WIN) / log (MIN_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE
  - F0K_WIN_DIFF_LOLO_NG = (log (MIN_STYLFIT_F0_WIN) / log (MIN_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE
  - F0K_WIN_DIFF_LOHI_NG = (log (MIN_STYLFIT_F0_WIN) / log (MAX_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE
  - F0K_WIN_DIFF_MNMN_NG = (log (MEAN_STYLFIT_F0_WIN) / log (MEAN_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE
- Difference and log difference between the last, mean, and minimum of the stylized F_0 values in a window and the baseline of F_0 values:
  - F0K_DIFF_LAST_KBASELN = LAST_STYLFIT_F0 - SPKR_FEAT_F0_BASELN
  - F0K_DIFF_MEAN_KBASELN = MEAN_STYLFIT_F0 - SPKR_FEAT_F0_BASELN
  - F0K_DIFF_WINMIN_KBASELN = MIN_STYLFIT_F0_WIN - SPKR_FEAT_F0_BASELN

- F0K_LR_LAST_KBASELN = log (LAST_STYLFIT_F0 / SPKR_FEAT_F0_BASELN)
- F0K_LR_MEAN_KBASELN = log (MEAN_STYLFIT_F0 / SPKR_FEAT_F0_BASELN)
- F0K_LR_WINMIN_KBASELN = log (MIN_STYLFIT_F0_WIN / SPKR_FEAT_F0_BASELN)

where LAST_STYLFIT_F0, MEAN_STYLFIT_F0 and MIN_STYLFIT_F0 are F_0 features.

- Normalization of the mean of the stylized F_0 values in the word (and next word) using the baseline, topline and range of F_0 values:
  - F0K_ZRANGE_MEAN_KBASELN = (MEAN_STYLFIT_F0 - SPKR_FEAT_F0_BASELN) / SPKR_FEAT_F0_RANGE
  - F0K_ZRANGE_MEAN_KTOPLN = (SPKR_FEAT_F0_TOPLN - MEAN_STYLFIT_F0) / SPKR_FEAT_F0_RANGE
  - F0K_ZRANGE_MEANNEXT_KBASELN = (MEAN_STYLFIT_F0_NEXT - SPKR_FEAT_F0_BASELN) / SPKR_FEAT_F0_RANGE
  - F0K_ZRANGE_MEANNEXT_KTOPLN = (SPKR_FEAT_F0_TOPLN - MEAN_FEAT_F0_NEXT) / SPKR_FEAT_F0_RANGE
- Difference and log difference between the mean and maximum of the stylized F_0 values (in the next word) and the topline of F_0 values:
  - F0K_DIFF_MEANNEXT_KTOPLN = MEAN_STYLFIT_F0_NEXT - SPKR_FEAT_F0_TOPLN
  - F0K_DIFF_MAXNEXT_KTOPLN = MAX_STYLFIT_F0_NEXT - SPKR_FEAT_F0_TOPLN
  - F0K_DIFF_WINMAXNEXT_KTOPLN = MAX_STYLFIT_F0_NEXT_WIN - SPKR_FEAT_F0_TOPLN
  - F0K_LR_MEANNEXT_KTOPLN = log (MEAN_STYLFIT_F0_NEXT / SPKR_FEAT_F0_TOPLN)
  - F0K_LR_MAXNEXT_KTOPLN = log (MAX_STYLFIT_F0_NEXT / SPKR_FEAT_F0_TOPLN)
  - F0K_LR_WINMAXNEXT_KTOPLN = log (MAX_STYLFIT_F0_NEXT_WIN / SPKR_FEAT_F0_TOPLN)
- Normalization of the maximum of the stylized F_0 values in the word (and next word) using the pitch mode and pitch of F_0 values:
  - F0K_MAXK_MODE_N = log (MAX_STYLFIT_F0 / SPKR_FEAT_F0_MODE)
  - F0K_MAXK_NEXT_MODE_N = log (MAX_STYLFIT_F0_NEXT / SPKR_FEAT_F0_MODE)
  - F0K_MAXK_MODE_Z = (MAX_STYLFIT_F0 - SPKR_FEAT_F0_MODE) / SPKR_FEAT_F0_RANGE
  - F0K_MAXK_NEXT_MODE_Z = (MAX_STYLFIT_F0_NEXT - SPKR_FEAT_F0_MODE) / SPKR_FEAT_F0_RANGE

  where MAX_STYLFIT_F0 and MAX_STYLFIT_F0_NEXT are F_0 features.
- Log difference between the stylized F_0 values in the word extremes:
  - F0K_WORD_DIFF_BEGBEG = log (FIRST_STYLFIT_F0 / FIRST_STYLFIT_F0_NEXT)
  - F0K_WORD_DIFF_ENDBEG = log (LAST_STYLFIT_F0 / FIRST_STYLFIT_F0_NEXT)
  - F0K_INWRD_DIFF = log (FIRST_STYLFIT_F0 / LAST_STYLFIT_F0)

  where FIRST_STYLFIT_F0, LAST_STYLFIT_F0 and FIRST_STYLFIT_F0_NEXT are F_0 features.

- Slope patterns and the normalization:
  - LAST_SLOPE: The last f or r in the F_0 feature PATTERN_SLOPE.
  - FIRST_SLOPE_NEXT: The first f or r in the F_0 feature PATTERN_SLOPE_NEXT.
  - SLOPE_DIFF_N = SLOPE_DIFF / SKPR_FEAT_F0_SD_SLOPE
  - LAST_SLOPE_N = LAST_SLOPE/ LAST_STYLFIT_F0

  where SLOPE_DIFF, LAST_DIFF and LAST_STYLFIT_F0 are F_0 features and SPKR_FEAT_F0_SD_SLOPE is obtained from table *spkr_feat.stats* (column STDEV_SLOPE) for the line corresponding to SPKR.

A.4.6 Energy Derived Features

The derived energy features are computed similarly as the derived F_0 features. The following is a list of the derived energy features.

- ENERGY_WORD_DIFF_HIHI_N
- ENERGY_WORD_DIFF_HILO_N
- ENERGY_WORD_DIFF_LOLO_N
- ENERGY_WORD_DIFF_LOHI_N
- ENERGY_WORD_DIFF_MNMN_N
- ENERGY_WORD_DIFF_HIHI_NG
- ENERGY_WORD_DIFF_HILO_NG
- ENERGY_WORD_DIFF_LOLO_NG
- ENERGY_WORD_DIFF_LOHI_NG
- ENERGY_WORD_DIFF_MNMN_NG
- ENERGY_WIN_DIFF_HIHI_N
- ENERGY_WIN_DIFF_HILO_N
- ENERGY_WIN_DIFF_LOLO_N
- ENERGY_WIN_DIFF_LOHI_N
- ENERGY_WIN_DIFF_MNMN_NG
- ENERGY_WIN_DIFF_HIHI_NG
- ENERGY_WIN_DIFF_HILO_NG
- ENERGY_WIN_DIFF_LOLO_NG
- ENERGY_WIN_DIFF_LOHI_NG
- ENERGY_WIN_DIFF_MNMN_NG

- ENERGY_DIFF_LAST_KBASELN
- ENERGY_DIFF_MEAN_KBASELN
- ENERGY_DIFF_WINMIN_KBASELN
- ENERGY_LR_LAST_KBASELN
- ENERGY_LR_MEAN_KBASELN
- ENERGY_LR_WINMIN_KBASELN
- ENERGY_ZRANGE_MEAN_KBASELN
- ENERGY_ZRANGE_MEAN_KTOPLN
- ENERGY_ZRANGE_MEANNEXT_KBASELN
- ENERGY_ZRANGE_MEANNEXT_KTOPLN
- ENERGY_DIFF_MEANNEXT_KTOPLN
- ENERGY_DIFF_MAXNEXT_KTOPLN
- ENERGY_DIFF_WINMAXNEXT_KTOPLN
- ENERGY_LR_MEANNEXT_KTOPLN
- ENERGY_LR_MAXNEXT_KTOPLN
- ENERGY_LR_WINMAXNEXT_KTOPLN
- ENERGY_MAXK_MODE_N
- ENERGY_MAXK_NEXT_MODE_N
- ENERGY_MAXK_MODE_Z
- ENERGY_MAXK_NEXT_MODE_Z
- ENERGY_WORD_DIFF_BEGBEG
- ENERGY_WORD_DIFF_ENDBEG
- ENERGY_INWRD_DIFF
- ENERGY_LAST_SLOPE
- ENERGY_SLOPE_DIFF_N
- ENERGY_LAST_SLOPE_N

A.4.7 Average Phone Duration

- AVG_PHONE_DUR_Z = $\sum_{every_phone_in_word}$phone_z[phone] / #phones
- MAX_PHONE_DUR_Z = $\max_{every_phone_in_word}$phone_z[phone]
- AVG_PHONE_DUR_N = $\sum_{every_phone_in_word}$phone_n[phone] / #phones
- MAX_PHONE_DUR_N = $\max_{every_phone_in_word}$phone_n[phone]

where

- #phones is the number of phones in the word
- phone_z[phone] =
  (phone_dur[phone] - phone_dur_mean[phone]) / phone_dur_stdev[phone]
- phone_n[phone] = phone_dur[phone] / phone_dur_mean[phone]
- phone_dur[phone] is the phone duration for phone (obtained from the feature WORD_PHONES) and phone_dur_mean[phone] and phone_dur_stdev[phone] are taken from table *phone_dur.stats*.

A.4.8 Speaker Specific Normalization

- AVG_PHONE_DUR_ZSP = $\sum_{every_phone_in_word}$phone_zsp[phone] / #phones
- MAX_PHONE_DUR_ZSP = $\max_{every_phone_in_word}$phone_zsp[phone]
- AVG_PHONE_DUR_NSP = $\sum_{every_phone_in_word}$phone_nsp[phone] / #phones
- MAX_PHONE_DUR_NSP = $\max_{every_phone_in_word}$phone_nsp[phone]

where

- #phones is the number of phones in the word
- phone_zsp[phone] =
  (phone_dur[phone] - spkr_phone_dur_mean[phone]) / spkr_phone_dur_stdev[phone]
- phone_nsp[phone] = phone_dur[phone] / spkr_phone_dur_mean[phone]
- phone_dur[phone] is the phone duration for phone (obtained from the feature WORD_PHONES). spkr_phone_dur_mean[phone] and spkr_phone_dur_stdev[phone] are taken from table *spkr_phone_dur.stats*.

Below are the features that are similar to the *_PHONE_DUR_* features except these are only over the vowels (not over every phone) in the word.

- AVG_VOWEL_DUR_Z
- MAX_VOWEL_DUR_Z

- AVG_VOWEL_DUR_N
- MAX_VOWEL_DUR_N
- AVG_VOWEL_DUR_ZSP
- MAX_VOWEL_DUR_ZSP
- AVG_VOWEL_DUR_NSP
- MAX_VOWEL_DUR_NSP

Bibliography

[1] J. Ang, Y. Liu, and E. Shriberg. Automatic dialog act segmentation and classification in multi-party meetings. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, March 2005.

[2] P. Boersma and D. Weeninck. Praat, a system for doing phonetics by computer. Technical Report 132, University of Amsterdam, Inst. of Phonetic Sc., 1996.

[3] L. Ferrer. Prosodic features extraction. Technical report, SRI, 2002.

[4] Y. Gotoh and S. Renals. Sentence boundary detection in broadcast speech transcript. In *Proc. of the Intl. Speech Communication Association (ISCA) Workshop: Automatic Speech Recognition: Challenges for the new Millennium ASR-2000*, 2000.

[5] Z. Huang, L. Chen, and M. Harper. An open source prosodic feature extraction tool. In *LREC006*, 2006.

[6] Z. Huang, L. Chen, and M. Harper. *Purdue Prosodic Feature Extraction Toolkit on Praat.* Spoken Language Processing Lab, Purdue University, ftp://ftp.ecn.purdue.edu/harper/praat-prosody.tar.gz, March 2006.

[7] Y. Liu, N. V. Chawla, M. P. Harper, E. Shriberg, and A. Stolcke. A study in machine learning from imbalanced data for sentence boundary detection in speech. *To appear in Computer Speech and Language*, 2005.

[8] Y. Liu, E. Shriberg, A. Stolcke, and M. Harper. Comparing HMM, maximum entropy, and conditional random fields for disfluency detection. In *INTERSPEECH*, Lisbon Spain, September 2005.

[9] Y. Liu, A. Stolcke, E. Shriberg, and M. Harper. Comparing and combining generative and posterior probability models: Some advances in sentence boundary detection in speech. In *Proceedings of the Empirical Methods in Natural Language Processing*, 2004.

[10] B. Pellom. SONIC: The University of Colorado continuous speech recognizer. Technical Report TR-CSLR-2001-01, University of Colorado, 2001.

[11] E. Shriberg and A. Stolcke. Direct modeling of prosody: An overview of applications in automatic speech processing. In *International Conference on Speech Prosody*, 2004.

[12] K. Sonmez, E. Shriberg, L. Heck, and M. Weintraub. Modeling dynamic prosodic variation for speaker verification. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 3189–3192, 1998.

[13] R. Sundaram, A. Ganapathiraju, J. Hamaker, and J. Picone. ISIP 2000 conversational speech evaluation system. In *Speech Transcription Workshop 2001*, College Park, Maryland, May 2000.

[14] C. Wightman and D. Talkin. *The Aligner*. Entropic, July 1997.