An Open Source Prosodic Feature Extraction Tool

Zhongqiang Huang, Lei Chen, Mary Harper

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
{huang37, chenl, harper}@ecn.purdue.edu

Abstract

There has been an increasing interest in utilizing a wide variety of knowledge sources in order to perform automatic tagging of speech events, such as sentence boundaries and dialogue acts. In addition to the words spoken, the prosodic content of the speech can be quite valuable for accurate event tagging. Shriberg and Stolcke (2004) have pioneered the “direct modeling” approach to exploit prosodic information in a variety of spoken language processing tasks such as sentence segmentation and tagging (Liu et al., 2004; Liu et al., 2005a), disfluency detection (Liu et al., 2005b), dialog act segmentation and tagging (Ang et al., 2005), and speaker recognition (Sonmez et al., 1998). An advantage of this approach is that no hand segmentation or intermediate labeling of the prosody 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 (Gotoh and Renals, 2000). By contrast, the above direct modeling efforts utilize a large number of features extracted using a proprietary prosodic feature extraction suite developed at SRI (Ferrer, 2002) to good effect. SRI’s feature extraction tool is Unix-based, combining ESPS/Waves for basic prosodic analysis (e.g., preliminary pitch tracking and energy computation (getF0)) with additional software components, such as a piecewise linear model (Sonmez et al., 1998) for pitch stylization. In this paper, we describe and evaluate an open source prosodic feature extraction tool1 based on Praat (Boersma and Weenink, 1996) 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.

Figure 1 illustrates how our prosodic feature extraction tool is used by an event detection system. As illustrated in Figure 1 (a), the goal of an event detection system is to determine at a particular point in the speech (usually a word boundary given forced alignment) whether to label it with an event (e.g., sentence unit boundary) or not. Figure 1 (b) shows an event tagging system that combines a language model (LM) with a prosody model, whose features are provided by our prosodic feature extraction system shown in Figure 1 (c). The rest of the paper is organized as follows. We briefly describe the prosodic features implemented in our tool in section 2, and present the implementation details and discuss some potential extension directions in section 3. In section 4, we evaluate the tool on the sentence boundary detection task using the NIST RT04 CTS data. The last section summarizes this work.

1. Introduction

Recently, there has been an increasing interest in utilizing a wide variety of knowledge sources in order to label speech with event tags, such as sentence boundaries and dialogue acts. In addition to the words spoken, the prosodic content of the speech can be quite valuable for accurate event tagging. Shriberg and Stolcke (2004) have pioneered the “direct modeling” approach to exploit prosodic information in a variety of spoken language processing tasks such as sentence segmentation and tagging (Liu et al., 2004; Liu et al., 2005a), disfluency detection (Liu et al., 2005b), dialog act segmentation and tagging (Ang et al., 2005), and speaker recognition (Sonmez et al., 1998). An advantage of this approach is that no hand segmentation or intermediate labeling of the prosody 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 (Gotoh and Renals, 2000). By contrast, the above direct modeling efforts utilize a large number of features extracted using a proprietary prosodic feature extraction suite developed at SRI (Ferrer, 2002) to good effect. SRI’s feature extraction tool is Unix-based, combining ESPS/Waves for basic prosodic analysis (e.g., preliminary pitch tracking and energy computation (getF0)) with additional software components, such as a piecewise linear model (Sonmez et al., 1998) for pitch stylization. In this paper, we describe and evaluate an open source prosodic feature extraction tool1 based on Praat (Boersma and Weenink, 1996) 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.

2. Feature Description

The features described here have been inspired by SRI’s prosodic feature extraction suite (Ferrer, 2002); however, our implementation differs in that we utilize the functionality of Praat. For example, pitch tracking is performed by Praat’s autocorrelation based pitch tracking function rather than using ESPS/Waves, and the pitch stylization is accomplished by Praat’s pitch stylization function, rather than the piecewise linear stylization algorithm in SRI’s model. Since we have been focusing on various types of boundary detection tasks, all of the prosodic features are extracted around each word boundary.

- Duration features: Duration features are obtained based on the word and phone alignments of human transcriptions (or ASR output). Pause duration and its normalization after each word boundary are extracted. We also measure the duration of the last vowel and the last rhyme, as well as their normalizations, for each word preceding a boundary. The duration and the normalized duration of each word are also included as duration features.

- F0 features: Praat’s autocorrelation based pitch tracker is used to obtain raw pitch values. The pitch

---

1This tool can be downloaded at ftp://ftp.ecn.purdue.edu/ harper/praat-prosody.tar.gz, along with a manual (Huang et al., 2006).
baseline and topline, as well as the pitch range, are computed based on the mean and variance of the logarithmic $F_0$ values. Voiced/unvoiced (VUV) regions are identified and the original pitch contour is stylized over each voiced segment. Several different types of $F_0$ features are computed based on the stylized pitch contour.

- **Range features**: These features reflect the pitch range of a single word or a window preceding or following a word boundary. These include the minimum, maximum, mean, and last $F_0$ values of a specific region (i.e., within a word or window) relative to each word boundary. These features are also normalized by the baseline $F_0$ values, the topline $F_0$ values, and the pitch range using linear difference and log difference.

- **Movement features**: These features measure the movement of the $F_0$ contour for the voiced regions of the word or window preceding and the word or window following a boundary. The minimum, maximum, mean, the first, and the last stylized $F_0$ values are computed and compared to that of the following word or window, using log difference and log ratio.

- **Slope features**: Pitch slope is generated from the stylized pitch values. The last slope value of the word preceding a boundary and the first slope value of the word following a boundary are computed. We also include the slope difference and dynamic patterns (i.e., falling, rising, and unvoiced) across a boundary as slope features, since a continuous trajectory is more likely to correlate with non-boundaries; whereas, a broken trajectory tends to indicate a boundary of some type.

- **Energy features**: The energy features are computed based on the intensity contour produced by Praat. Similar to the $F_0$ features, a variety of energy related range features, movement features, and slope features are computed, using various normalization methods.

- **Other features**: We add the gender type to our feature set. Currently the gender information is provided in
3. Implementation and Extension

We have implemented the prosodic feature extraction tool using Praat’s programmable scripting language (Boersma and Weenink, 1996). A very important reason that we chose Praat as our platform is that it provides an existing suite of high quality speech analysis routines, such as pitch tracking. Additional reasons for using Praat include:

1. Praat is a public domain, widely used speech analysis toolkit that is supported on a variety of platforms (e.g., Windows, Macintosh, Linux, and Solaris).

2. It provides a variety of valuable data structures, such as TextGrid, PitchTier, and Table, to represent various types of information used for extracting prosodic features.

3. It provides a built-in programmable scripting language for calling Praat’s commands and extending its capability.

4. Additions to Praat functionality can be immediately adopted into the tool. This is especially useful for incorporating new prosodic features.

Hence, Praat is an ideal platform for building a public domain prosodic feature extraction tool that can be used and extended by a wide variety of researchers. In the rest of this section, we describe our tool implementation and discuss some potential ways in which the tool could be extended.

3.1. Implementation

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, VUV) representing duration, \( F_0 \), and energy information, as is shown in Figure 1 (c). 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 (Ferrer, 2002) and our tool documentation (Huang et al., 2006). We describe below how to obtain and represent these basic elements in Praat. Table 1 summarizes their use in the computation of the prosodic features.

- **Word and Phone Alignments**: A forced alignment system\(^2\) is used to determine the starting and ending times of words and phones. In our tool these alignments are represented in TextGrid IntervalTier.

- **Vowel and Rhyme**: The starting and ending times of vowels and rhymes are determined from the phone alignments. As to rhyme, we only consider the last rhyme, which is defined to be the sequence of phones starting from the last vowel and covering all the remaining phones in a word. Vowels and rhymes are also represented in TextGrid IntervalTier.

- **\( F_0 \)**: We rely on Praat’s autocorrelation based pitch tracking algorithm to extract raw pitch values, using gender dependent pitch range. The raw pitch contour is smoothed and the voiced/unvoiced regions are determined and stored in a TextGrid IntervalTier. Praat’s pitch stylization function is used to stylize raw \( F_0 \) values over each voiced region. Both raw \( F_0 \) values and stylized \( F_0 \) values are represented in PitchTier. The pitch slope values are generated based on the stylized pitch contour, and are stored in a TextGrid IntervalTier.

- **Energy**: Intensity values are computed for each frame and stored in an IntensityTier. Since there is no intensity stylization function in Praat, we choose to represent intensity values in a PitchTier, and apply the pitch stylization function to stylize the intensity contour. Note that stylization is performed on the entire intensity contour, in contrast to the pitch case, for which this applies only in voiced regions. The raw and stylized intensity values are stored in PitchTier, and the slope values are stored in a TextGrid IntervalTier.

### Table 1: The use of raw files for extracting various features.

| Features          | Duration | \( F_0 \) | Energy |
|-------------------|----------|-----------|--------|
| Word              | \checkmark | \checkmark | \checkmark |
| Phone             | \checkmark | \xmark    | \xmark |
| Vowel             | \checkmark | \xmark    | \xmark |
| Rhyme             | \xmark    | \checkmark | \xmark |
| VUV               | \xmark    | \checkmark | \xmark |
| Raw Pitch         | \xmark    | \checkmark | \xmark |
| Stylized Pitch    | \xmark    | \checkmark | \xmark |
| Pitch Slope       | \xmark    | \checkmark | \xmark |
| Raw Energy        | \xmark    | \xmark    | \checkmark |
| Stylized Energy   | \xmark    | \xmark    | \checkmark |
| Energy Slope      | \xmark    | \xmark    | \checkmark |

\(^2\)This is separate from our feature extraction tool. Researchers can choose from a variety of systems, such as Aligner (Wightman and Talkin, 1997), ISIP ASR (Sundaram et al., 2000), and SONIC (Pellom, 2001).
Take jitter and shimmer for example. Jitter is the cycle-to-cycle variability in frequency of vocal fold vibration. Shimmer is the cycle-to-cycle variability in amplitude of the vocal fold vibration. These two measurements have been investigated as voice quality measurements for pathological speech. Liu (2003) used a variety of prosodic features (i.e., duration, pitch, and energy features) together with some voice quality measures for word fragment identification in conversational speech. She found that jitter was the most frequently queried voice quality feature in the decision tree created for the task. There are existing algorithms in Praat for computing both jitter and shimmer; therefore, it would be a simple matter to implement a jitter and shimmer extraction script and add it into our tool. In fact, Liu obtained the jitter values from Praat.

Take Fujisaki analysis for another example. The Fujisaki model (Fujisaki and Kawai, 1988) is a parametric superpositional model widely used in the area of speech synthesis. In this model, the fundamental frequency contour is decomposed into phrase components and accent components over the base frequency. One can imagine that the phrase components are informative of phrase or sentence boundaries, thus could be used in some boundary detection tasks. Although Fujisaki analysis has yet to be implemented in Praat, if it were, it would be a simple matter to add a Fujisaki analysis into our tool in order to provide an additional set of features.

4. Evaluation

4.1. Setup

In order to examine the effectiveness of our tool, i.e., the quality of the features extracted by the tool on event tagging tasks, we evaluated the extracted features on SU detection task using the NIST RT04 CTS data. The RT04 CTS data has a training set drawn from the Switchboard corpus, two development sets with one drawn from both Switchboard and Fisher and another from Fisher, and an eval set drawn from Fisher. To evaluate the in-domain and out-of-domain effects, we have chosen to train our prosody model on the entire training set and evaluate on the Switchboard part of the first development set and the Fisher data in the eval set. Table 2 summarizes the data used in our experiments. SONIC’s aligner (Pellom, 2001) was used to produce the word and phone alignments given the word boundaries in the official NIST RT04 RTTM files as reference. Pronunciations were created for all out of vocabulary (OOV) words.

| #SU | #Words | Source |
|-----|--------|--------|
| train | 64K | 480K | Switchboard |
| dev | 6K | 35K | Switchboard |
| eval | 5K | 34K | Fisher |

Table 2: The size and source of experimental data.

A standard CART style decision tree was used to train the prosody models (an example tree is given in Figure 2). In order to address the imbalanced data problem (since there are fewer SU events than non-events at interword boundaries; overall only 13.3% word boundaries in the training set are also sentence boundaries), we used a downsampled training set in which SU and non-SU classes had equal prior probabilities. Additionally, we employed ensemble bagging to reduce the variance of the prosodic classifiers. Using this method, several random downsampled training sets were generated, and each was resampled multiple times and corresponding classifiers were combined via bagging. This has been found to improve the performance of the prosody model on the CTS SU detection task (Liu et al., 2005a; Liu, 2004).

Figure 2: An example of a decision tree for SU detection. Each line represents a node in the tree, with the associated question regarding a particular feature (e.g., PAUSE_DUR is the pause duration after a word boundary, F0_K_WIN_DIFF_LOGLO。<sub>LOG</sub> is the log ratio between the minimum of the stylized F0 within a window before a word boundary and the minimum of the stylized F0 within a window after a word boundary, normalized by the pitch range). The ‘0’ (non-SU) or ‘1’ (SU) at the end of a line (representing a leaf note) denotes the decision at the corresponding node.

To measure the performance of our models, we used the Error Rate metric defined by NIST for the DARPA EARS metadata evaluation for comparison with the literature. The Insertion Rate and Deletion Rate are also provided to determine whether there are different patterns of insertions and

---

3In our experiment, one conversation from Switchboard was eliminated from the dev set due to a transcription problem.

4The posterior probabilities provided by decision tree models are normalized to reflect the distribution of SU/non-SU boundaries in the training set.
deletions among different feature sets. The three metrics are defined below:

1. **Insertion Rate** = \#Insertions / \#SUs in the reference
2. **Deletion Rate** = \#Deletions / \#SUs in the reference
3. **Error Rate** = (\#Deletions + \#Insertions) / \#SUs in the reference

### 4.2. Results and Discussion

We started with all of the prosodic features (more than two hundred) extracted by our tool, and compared in Table 3 the performance of the decision tree model trained using these features to a simple baseline that always selected the majority class (i.e., non-SU). We also included for comparison the performance of using only pause duration (pause\_dur) after a word boundary. As can be observed, using the full set of prosodic features improves performance considerably over the chance baseline, and also reduces overall error rate compared to using pause duration alone. Although the improvement on the eval set using the full feature set against pause duration alone is greater than on the dev set, the prosody model has a much larger error than on the dev set, which was drawn from the Switchboard corpus. This may be caused by the fact that the eval set is entirely composed of Fisher data, and there are some differences between the Switchboard corpus and the Fisher corpus.

| Evaluation on dev set | Deletion | Insertion | Error |
|-----------------------|----------|-----------|-------|
| Baseline              | 100      | 0         | 100   |
| pause\_dur            | 41.72    | 17.75     | 59.47 |
| full set              | 51.45    | 5.88      | 57.33 |
| Evaluation on eval set| Deletion | Insertion | Error |
| Baseline              | 100      | 0         | 100   |
| pause\_dur            | 41.93    | 27.10     | 69.03 |
| full set              | 56.42    | 6.17      | 62.59 |

Table 3: SU detection results using the full set of prosodic features, pause duration only, and a baseline obtained by considering all word boundaries to be non-events.

As pointed out in (Shriberg et al., 2000), the full feature set contains a high degree of feature redundancy, and may result in a suboptimal tree due to the greedy nature of the decision tree. We found that many of the unnormalized \( F_0 \) and energy features were rarely queried in the final decision trees. Hence, in a second experiment, we started with about 100 features comprised of most of the duration features, the normalized \( F_0 \) features, the normalized energy features, and gender type, and tested different combinations of these features (with gender type always included). The results are provided in Table 4.

| Evaluation on dev set | Deletion | Insertion | Error |
|-----------------------|----------|-----------|-------|
| Baseline              | 100      | 0         | 100   |
| dur+P0                | 49.41    | 6.71      | 56.13 |
| dur+P0+energy         | 57.94    | 6.01      | 59.94 |
| dur+P0+energy         | 49.96    | 6.60      | 56.56 |
| Evaluation on eval set| Deletion | Insertion | Error |
| Baseline              | 100      | 0         | 100   |
| dur+P0                | 49.98    | 9.87      | 59.85 |
| dur+P0+energy         | 47.52    | 11.56     | 59.08 |
| dur+P0+energy         | 52.80    | 8.01      | 60.82 |
| dur+P0+energy         | 51.31    | 8.88      | 60.19 |

Table 4: SU detection results using different combinations of prosodic features.

We also observe in Table 4 that, combining energy features with the duration features impairs the performance somewhat compared to using duration features only (\( p < 0.013 \), using the sign test) and the eval set (\( p < 0.061 \), using the sign test). Different training and test data may account for the discrepancy; however, there are other factors that may take effect. First, our tool uses different pitch tracking and pitch stylization algorithms than those in SRI’s suite, and thus the \( F_0 \) features are somewhat different. Second, the accuracy of word/phone alignments may effect the quality of \( F_0 \) features greatly since all these features are computed around each word boundary. In our preliminary study, we found that, if the alignments were generated using only the reference transcription without referring to the reference word boundaries, then combining \( F_0 \) features with duration features worked slightly worse than using duration features only.

We also observe in Table 4 that, combining energy features with the duration features impairs the performance somewhat compared to using duration features only (\( p < 0.02 \), using the sign test), and the performance degradation of using dur+energy compared to using dur+P0 is even more significant (\( p < 0.0003 \), using the sign test). However, it is incorrect to conclude that certain features such as the energy-related features are not useful for several reasons. First, as explained by Shriberg and Stolcke (Shriberg et al., 2000), these features may be useful on their own, but less useful when combined with other features. Second, the effectiveness of a certain feature varies across different corpora and tasks. Shriberg and Stolcke (Shriberg et al., 2000) reported that \( F_0 \) features contributed 11% to feature usage\(^3\) in sentence segmentation using Broadcast News data and 36% for topic segmentation; whereas, \( F_0 \) features were not among the feature set identified by their feature selection algorithm in the sentence segmentation task using Switchboard data. Liu (2003) reported that both energy and \( F_0 \) features were effective for word fragment detection.

Our system has around a 60% Error Rate on the RT04

\(^3\)Feature usage (Shriberg et al., 2000) is a measure computed as the relative frequency with which that feature or feature class is queried in the decision tree.
evaluation set, which is comparable to Liu’s result for the RT03 evaluation set (Liu, 2004), in which the prosodic features were extracted by SRI’s suite. Rather than being used alone, a prosody model is usually integrated with a language model for better performance. To complete this evaluation, the best prosody model (PM) in Table 4 is combined with a language model6 (LM) trained on the training set of the RT04 CTS data. Results are given in Table 5. The integration of the prosody and language models significantly reduces the Deletion Rate compared to the prosody model alone, and the Insertion Rate and overall Error Rate compared to each of the models alone.

### Table 5: SU detection results combining a prosodic model with a language model.

|                  | Deletion | Insertion | Error |
|------------------|----------|-----------|-------|
| Baseline         | 100      | 0         | 100   |
| PM               | 47.08    | 12.63     | 43.84 |
| LM               | 31.21    | 12.63     | 43.84 |
| PM + LM          | 31.69    | 4.16      | 35.86 |

### Evaluation on dev set

|                  | Deletion | Insertion | Error |
|------------------|----------|-----------|-------|
| Baseline         | 100      | 0         | 100   |
| PM               | 47.52    | 11.56     | 59.08 |
| LM               | 30.83    | 14.65     | 45.48 |
| PM + LM          | 32.18    | 5.69      | 37.87 |

### Evaluation on eval set

5. Conclusion

In this paper, we have reported on our open source prosodic feature extraction tool built based on Praat. This tool is capable of extracting a variety of duration features, $F_0$ features, and energy features given an audio recording and its time aligned words and phones sequences. Implementation details are described and extension capabilities are discussed. Evaluation of the prosody model derived from this tool and its integration with a language model have demonstrated the effectiveness of our tool on the sentence boundary detection task using NIST RT04 CTS data.

In the future, we will apply this tool to other event tagging applications, and will also continue to expand the tool’s capability for the research communities.

6. Acknowledgments

This work was supported in part by ARDA under contract number MDA904-03-C-1788. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of ARDA. We would also like to acknowledge Yang Liu for sharing her expertise regarding prosodic features and evaluation experiments.

7. References

J. Ang, Y. Liu, and E. Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings.

---

6 We trained a 4-gram hidden event (i.e., SU and non-event) language model as in (Liu, 2004) with Kneser-Ney discounting and interpolation. Note that we did not normalize word tokens from the RT04 corpus for the language model because the original (unnormalized) word tokens from RT04 were used for training and testing the prosody model. Hence, to support model combination based on posterior probabilities, we utilized a slightly poorer LM that we could have constructed using normalized tokens.