Table 2. Predicted L1 group membership (percent correct) of five utterances according to a discriminant analysis using seven metrics (see section 2.2).

	Predicted L1 group membership						
L1 group	Chinese	English	French	German	Persian	Russian	Norwegian
Chinese	**80**	0	0	0	20	0	0
English	0	**100**	0	0	0	0	0
French	0	0	**80**	0	0	20	0
German	0	0	0	**100**	0	0	0
Persian	0	0	0	0	**100**	0	0
Russian	0	0	0	0	0	**100**	0
Norwegian	0	0	0	0	0	0	**100**

The results of the discriminant analysis further showed that three of the six discriminant functions reached statistical significance, cumulatively explaining 96.4% of the variance. For the first function, the metrics with most discriminatory power were slope (metric 4), speech rate (metric 5) and mean syllable duration (metric 1). The second discriminant function had also slope and speech rate, but additionally standard deviations for speech rate (metric 6) and for syllable duration (metric 2), and nPVI (metric 7) as important variables. Finally, of highest importance for the third function were metrics 5, 3 (correlation coefficient), 4, and 7, in that order.

4 Conclusion

The present results suggest that the utterances spoken by the second language users differed in rhythmical structure from those produced by the native speakers. It was shown that it is possible to quantify rhythm using direct and indirect measures. Though the statistical analysis yielded promising results, it should be kept in mind that the number of utterances investigated was relatively small. Therefore, more research will be needed to confirm the preliminary results and to refine the present approach.

Acknowledgements

This research is supported by the Research Council of Norway (NFR) through grant 158458/530 to the project Språkmøter. I would like to thank Rein Ove Sikveland for the segmentation of the speech material.

References

Barry, W.J., B. Andreeva, M. Russo, S. Dimitrova & T. Kostadinova, 2003. Do rhythm measures tell us anything about language type? *Proceedings 15th ICPhS*, Barcelona, 2693-2696.

Boersma, P. & D. Weenink, 2006. Praat: doing phonetics by computer (Version 4.4.11) [Computer program]. Retrieved February 23, 2006, from http://www.praat.org/.

Grabe, E. & E.L. Low, 2002. Durational variability in speech and the rhythm class hypothesis. In C. Gussenhoven & N. Warner (eds.), *Laboratory Phonology 7*. Berlin: Mouton, 515-546.

Ramus, F., 2002. Acoustic correlates of linguistic rhythm: Perspectives. *Proceedings Speech Prosody 2002*, Aix-en-Provence, 115-120.

Ramus, F., M. Nespor & J. Mehler, 1999. Correlates of linguistic rhythm in the speech signal. *Cognition 73,* 265-292.

Stockmal, V., D. Markus & D. Bond, 2005. Measures of native and non-native rhythm in a quantity language. *Language and Speech 48,* 55-63.

/nailon/ – Online Analysis of Prosody

Jens Edlund and Mattias Heldner
Department of Speech, Music and Hearing, KTH, Stockholm
{edlund|mattias}@speech.kth.se

Abstract

This paper presents `/nailon/` *– a software package for online real-time prosodic analysis that captures a number of prosodic features relevant for interaction control in spoken dialogue systems. The current implementation captures silence durations; voicing, intensity, and pitch; pseudo-syllable durations; and intonation patterns. The paper provides detailed information on how this is achieved.*

1 Introduction

All spoken dialogue systems, no matter what flavour they come in, need some kind of interaction control capabilities in order to identify places where it is legitimate to begin to talk to a human interlocutor, as well as to avoid interrupting the user. Most current systems rely *exclusively* on silence duration thresholds for making such interaction control decisions, with thresholds typically ranging from 500 to 2000 ms (e.g. Ferrer, Shriberg & Stolcke, 2002). Such an approach has obvious drawbacks. Users generally have to wait longer for responses than in human-human interactions, but at the same time they run the risk of being interrupted by the system. This is where **/nailon/** – our software for online analysis of prosody and the main focus of this paper – enters the picture.

2 Design criteria for practical applications

In order to use prosody in practical applications, the information needs to be available to the system, which places special requirements on the analyses. First of all, in order to be useful in live situations, all processing must be performed automatically, in real-time and deliver its results with minimal latency (cf. Shriberg & Stolcke, 2004). Furthermore, the analyses must be online in the sense of relying on past and present information only, and cannot depend on any right context or look-ahead. There are other technical requirements: the analyses should be sufficiently general to work for many speakers and many domains, and should be predictable and constant in terms of memory use, processor use, and latency. Finally, although not a strict theoretical nor a technical requirement, it is highly desirable to use concepts that are relevant to humans. In the case of prosody, measurements should be made on psychoacoustic or perceptually relevant scales.

3 **/nailon/**

The prosodic analysis software **/nailon/** was built to meet the requirements and to capture silence durations; voicing, intensity, and pitch; pseudo-syllable durations; and intonation patterns. It implements high-level methods accessible through in Tcl/Tk and the low-level audio processing is handled by the Snack sound toolkit, with pitch-tracking based on the ESPS tool get_f0. **/nailon/** differs from Snack in that its analyses are incremental with relatively small footprints and can be used for online analyses. The implementation is real-time in the sense that it performs in real time, with small and constant latency, on a standard PC. It is a key feature that the processing is online – in fact, **`/nailon/`** is a phonetic anagram of online. On the acoustic level, this goes well with human circumstances as humans rarely need acoustic right context to make decisions about segmentation. The requirements on memory and processor usage are met by using incremental algorithms, resulting in a system with a small and constant footprint and flexible processor usage. The generality requirements are met by using online normalisation and by avoiding algorithms relying on ASR. The analysis is in some ways similar to that used by Ward & Tsukahara (2000), and is performed in several consecutive steps. Each step is described in detail below.

3.1 Audio acquisition

The audio signal is acquired through standard Snack object methods from any audio device. Each new frame is pushed onto a fixed length buffer of predetermined size, henceforth the *current buffer*. The buffer size is a factor of the *processing unit size*. Note that processing unit size is not the inverse of the sampling frequency, which defaults to 1/16 kHz. Rather, it should be larger by an order of magnitude to ensure smooth processing. The default processing unit size is 10 ms, and the default current buffer size is 40 such units, or 400 ms. The current buffer, then, is in effect a moving window with a length of less than half a second. As far as the processing goes, sound that is pushed out on the left side of the buffer is lost, as the Snack object used for acquisition is continuously truncated. The current buffer is updated with every time a sufficient sound to fill another processing unit has been acquired – 100 times per second given the default settings.

3.2 Preprocessing

In many cases, the online requirement makes it impractical or impossible to use filters directly on the Snack sound object used for acquisition. Instead, **`/nailon/`** provides access to the raw current audio buffer, so that filters can be applied to it before any other processing takes place. Filters are applied immediately before each get_f0 extraction (see the next section). Using filters in this manner causes **`/nailon/`** to duplicate the current audio buffer in order to have a raw, unfiltered copy of the buffer available at all times.

3.3 Voicing, pitch, and intensity extraction

Voicing, pitch, and intensity are extracted from the current buffer using the Snack/ESPS get_f0 function. This process is made incremental by repeating it over the current buffer as the buffer is updated. The rate at which extraction takes place is managed externally, which facilitates robust handling of varying processor load caused by other processes. In an ideal situation, the update takes place every time a new processing unit has been pushed onto the current buffer, in which case only the get_f0 results for the very last processing unit of the buffer are used. If this is not possible due to processor load, then a variable number *N* processing units will have been added to the buffer since the last F0 extraction took place, and the last *N* results from get_f0 will be used, where *N* is a number smaller then the length of the current buffer processing units. In this case, we introduce a latency of *N* processing units to the processing at this stage. **`/nailon/`** configuration permits that a maximum update rate is given in order to put a cap on the process requirements of the analysis. The default setting is to process every time a single processing unit has been added, which provides smooth processing on a regular PC at a negligible latency. Each time a get_f0 extraction is performed, **`/nailon/`** raises an event for each of the new get_f0 results produced by the extraction, in sequence. These events, called *ticks*, trigger each of the following processing steps.

3.4 Filtering

Each tick triggers a series of event driven processing steps. These steps are generally optional and can be disabled to save processing time. The steps described here are the ones used by default. The first step is a filter containing a number of reality checks. Pitch and intensity are checked against preset minimum and maximum thresholds, and set to an undefined value if they fail to meet these. Similarly, if voicing was detected, this is removed if the pitch is out of bounds. Correction for octave errors is planned to go here as well, but not currently implemented. Note that removing values at this stage does not put an end to further processing – consecutive processes may continue by extrapolation or other means. Median filtering can be applied to the pitch and intensity data. If this is done, a delay of half the total time of the number of processing units used in the filtering is introduced at this point. By default, a median filter of seven processing units is used. In effect, this causes all consecutive processes to focus on events that took place 3 processing units back, causing a delay of less than 40 ms. On the other hand, the filter makes the analysis more robust. Finally, the resulting pitch and intensity values are transformed into semitones and dB, respectively.

3.5 Range normalisation of F0 and intensity

`/nailon/` implements algorithms for calculation of incremental means and standard deviations. Each new processing unit causes an update in mean and standard deviation of both pitch and intensity, provided that it was judged to contain voiced speech by the previous processing stage. The dynamic mean and standard deviation values are used as a model for normalising and categorising new values. The stability of the model is tracked by determining whether the standard deviation is generally decreasing. Informal studies show that the model stabilises after less than 20 seconds of speech has been processed, given a single speaker in a stable sound environment. Currently, **`/nailon/`** may either cease updating means and standard deviation when stability is reached, or continue updating them indefinitely, with ever-decreasing likelihood that they will change. A possibility to reset the model is also available. A decaying algorithm which will permit us to fine-tune how stable or dynamic the normalisation should be has been designed, but has yet to be implemented. The mean and standard deviation are used to normalise the values for pitch and intensity with regard to the preceding speech by expressing them as the distance from the mean expressed in standard deviations.

3.6 Silence detection

Many of the analyses we have used **`/nailon/`** for to date are refinements or additions of speech/silence decisions. For this reason a simplistic speech activity detection (SAD) is implemented. Note, however, that **`/nailon/`** would work equally well or better together with an external SAD. **`/nailon/`** uses a simple intensity threshold which is recalculated continuously and is defined as the valley following the first peak in an intensity histogram. **`/nailon/`** signals a change from silence to speech whenever the threshold is exceeded for a configurable number of consecutive processing units, and vice versa. The default number is 30, resulting in a latency of 300 ms for speech/silence decisions. Informal tests show no decrease in performance if this number is lowered to 20, but it should be noted that the system has only been used on sound with a very good signal-to-noise ratio.

3.7 Psyllabification

`/nailon/` keeps a copy of pitch, intensity and voicing information for the last seen consecutive stretch of speech at all times. Whenever silence is encountered, this intensity values of the record are searched backwards (last processing unit first) for a *convex hull* (loosely based on Mermelstein, 1975) contained in it. A hull in the intensity values of speech is assumed to correspond roughly to a syllable, thus providing a pseudo-syllabification, or *psyllabification*. By searching backwards, the hull that occurred last is found first. Currently, processing ceases at this point, since only the hulls directly preceding silence has been of interest to us so far. A convex hull in **/nailon/** is defined as a stretch of consecutive value triplets ordered chronologically, where the centre value is always above or on a line drawn between the first and the last value. As this definition is very sensitive to noisy data, it is relaxed by allowing a limited number of values to drop below the line between first and last value as long as the area between that line and the actual values is less than a preset threshold.

3.8 Classification

The normalised pitch, intensity, and voicing data extracted by **/nailon/** over a *psyllable* are intended for classification of intonation patterns. Each silence-preceding hull is classified into HIGH, MID, or LOW depending on whether the pitch value is in the upper, mid or lower third of the speaker's F0 range described by mean and standard deviation, and into RISE, FALL, and or LEVEL depending on the shape of the intonation pattern. Previous work have shown that the prosodic information provided by **/nailon/** can be used to improve the interaction control in spoken human-computer dialogue compared to systems relying exclusively on silence duration thresholds (Edlund & Heldner, 2005).

4 Discussion

In this paper, we have presented **/nailon/**, an online, real-time software package for prosodic analysis capturing a number of prosodic features liable to be relevant for interaction control. Future work will include further development of **/nailon/** in terms of improving existing algorithms – in particular the intonation pattern classification – as well as adding new prosodic features. For example, we are considering evaluating the duration of psyllables as an estimate of final lengthening or speaking rate effects, and to use intensity measures to capture the different qualities of silent pauses resulting from different vocal tract configurations (Local & Kelly, 1986).

Acknowledgements

This work was carried out within the CHIL project. CHIL is an Integrated Project under the European Commission's sixth Framework Program (IP-506909).

References

Edlund, J. & M. Heldner, 2005. Exploring Prosody in Interaction Control. *Phonetica 62*, 215-226.

Ferrer, L., E. Shriberg & A. Stolcke, 2002. Is the speaker done yet? Faster and more accurate end-of-utterance detection using prosody in human-computer dialog. *Proceedings ICSLP 2002*, Denver, 2061-2064.

Local, J.K. & J. Kelly, 1986. Projection and 'silences': Notes on phonetic and conversational structure. *Human Studies 9*, 185-204.

Mermelstein, P., 1975. Automatic segmentation of speech into syllabic units. *Journal of the Acoustical Society of America 58*, 880-883.

Shriberg, E. & A. Stolcke, 2004. Direct Modeling of Prosody: An Overview of Applications in Automatic Speech Processing. *Proceedings Speech Prosody 2004*, Nara, 575-582.

Ward, N. & W. Tsukahara, 2000. Prosodic features which cue back-channel responses in English and Japanese. *Journal of Pragmatics 32*, 1177-1207.

Lund University, Centre for Languages & Literature, Dept. of Linguistics & Phonetics
Working Papers 52 (2006), 41–44

Feedback from Real & Virtual Language Teachers

Olov Engwall
Centre for Speech Technology, KTH
engwall@kth.se

Abstract

Virtual tutors, animated talking heads giving the student computerized training of a foreign language, may be a very important tool in language learning, provided that the feedback given to the student is pedagogically sound and effective. In order to set up criteria for good feedback from a virtual tutor, human language teacher feedback has been explored through interviews with teachers and students, and classroom observations. The criteria are presented together with an implementation of some of them in the articulation tutor ARTUR.

1 Introduction

Computer assisted pronunciation training (CAPT) may contribute significantly to second language learning, as it gives the students access to private training sessions, without time constraints or the embarrassment of making errors in front of others. The success of CAPT is nevertheless still limited. One reason is that the detection of mispronunciations is error-prone and that this leads to confusing feedback, but Neri et al. (2002) argue that successful CAPT is already possible, as the main flaw lies in the lack of pedagogy in existing CAPT software rather than in technological shortcomings. They conclude that if only the learners' needs, rather than technological possibilities, are put into focus during system development, pedagogically sound CAPT could be created with available technology.

One attempt to answer this pedagogical need is to create virtual tutors, computer programs where talking head models interact as human language teachers. An example of this is ARTUR – the ARticulation TUtoR (Bälter et al., 2005), who gives detailed audiovisual instructions and articulatory feedback. Refer to www.speech.kth.se/multimodal/ARTUR for a video presenting the project. In such a virtual tutor system it becomes important not only to improve the pedagogy of the given feedback, but to do it in such a way that it resembles *human* feedback, in order to take benefit of the social process of learning

To test the usability of the system at an early stage, we are conducting Wizard of Oz studies, in which a human judge detects the mispronunciations, diagnoses the cause and chooses what feedback ARTUR should give from a set of pre-generated audiovisual instructions (Bälter et al., 2005). The children practicing with ARTUR did indeed like it, but the feedback was sometimes inadequate, e.g. when the child repeated the same error several times; when the error was of the same type as before, but the pronunciation had been improved, or when the student started to loose motivation, because the virtual tutor's feedback was too detailed. One conclusion was hence that a more varied feedback was needed in order to be efficient. The aim of this study is to investigate how the feedback of the virtual tutor could be improved by studying feedback strategies of human language teachers in pronunciation training and assess which of them could be used in ARTUR. Interviews with language teachers and students, and classroom observations were used to explore *when* feedback should be given, *how* to indicate an error, *which errors* should be corrected, and how to *promote student motivation*.