/nailon/ – Software for Online Analysis of Prosody

Jens Edlund & Mattias Heldner

Department of Speech, Music and Hearing
KTH, Stockholm, Sweden
[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. As an example application of **/nailon/**, we demonstrate how it is used to improve the efficiency of identifying relevant places at which a machine can legitimately begin to talk to a human interlocutor, as well as to shorten system response times. **Index Terms**: automatic extraction of prosodic features, dialogue systems, interaction control

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 [1, 2]. Such an approach has several drawbacks, both from the point of view of the user and that of the system. Users generally have to wait longer for responses than in human-human interactions; at the same time they run the risk of being interrupted by the system, since people frequently pause also when they are not done speaking, for example when hesitating or before semantically heavy words [2, 3]; and using silent pauses as the sole information for segmentation of user input is likely to impair the system's speech understanding, as unfinished or badly segmented utterances often are more difficult to interpret [4].

Humans are very good at discriminating the places where their conversational partners have finished talking from those where they have not – accidental interruptions are rare in conversations. Apparently, we use a variety of information to do so, including numerous prosodic and gestural features, as well as higher levels of understanding, for example related to (in)completeness on a structural level [e.g. 5, 6, 7].

In light of this, the interaction control capabilities of spoken dialogue systems would likely benefit from access to more of this variety of information – more than just the duration of silent pauses. Ultimately, spoken dialogue systems should of course be able to combine all relevant and available sources of information for making interaction control decisions. Attempts have been made at using semantic information [4, 8], prosodic information and in particular intonation patterns [1, 3, 9], and visual information [9] to deal with (among other things) the problems that occur as a result of interaction control decisions based on silence only. 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 additional information, including prosody, to improve interaction control in practical applications such as in spoken dialogue systems, the information needs to be made 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. 2]. 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 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. Prosodic cues for interaction control

Previous work suggests that a number of prosodic or phonetic cues are liable to be relevant for interaction control in human-human dialogue. Ultimately, software for improving interaction control in practical applications should capture all relevant cues.

The phenomena associated with turn-yielding include silent pauses, falling and rising intonation patterns, and certain vocal tract configurations such as exhalations [e.g. 5, 7, 10]. Turn-yielding cues are typically located somewhere towards the end of the contribution, although not necessarily on the final syllable. Granted that human turn-taking involve decisions above a reflex level, evidence suggest that turn-yielding cues must occur at least 200-300 ms before the onset of the next contribution [11].

The phenomena associated with turn-keeping include level intonation patterns, vocal tract configurations such as glottal or vocal tract stops without audible release, as well as a different quality of silent pauses as a result of these vocal tract closures [e.g. 5, 12, 13]. Turn-keeping cues are also located near the end of the contribution. As these cues are not

intended to trigger a response, but rather to inhibit one, they may conceivably occur later than turn-yielding cues.

There are also a number of cues (in addition to the silent pauses mentioned above) that have been observed to occur with turn-yielding as well as with turn-keeping. Examples of such cues include decreasing speaking rate and other lengthening patterns towards the end of contributions. The mere presence (or absence) of such cues cannot be used for making a turn-yielding vs. turn-keeping distinction, although the amount of final lengthening, for example, might provide valuable guidance for such a task [cf. 14].

4. /nailon/

The prosodic analysis software package **/nailon/** was built to meet the requirements and to capture some of the cues listed above. The current implementation captures silence durations; voicing, intensity, and pitch; pseudo-syllable durations; and intonation patterns. It implements high-level methods accessible through Tcl/Tk (http://www.tcl.tk/). The low-level audio processing is handled by the Snack sound toolkit (http://www.speech.kth.se/snack/) developed at KTH, with pitch-tracking based on the ESPS tool get_f0. Tcl/Tk as well as Snack are available for several platforms, including Windows, Linux and Mac OS X. Thus, **/nailon/** ought to run on all of these, but the current version has only been tested on Windows. **/nailon/** differs from Snack in that its analyses are incremental, making them suitable for online analyses. The implementation performs in real-time, with small and constant latency, footprint and processor usage, 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. Humans rarely need acoustic right context to make decisions about segmentation. On the contrary, they often seem to be able to predict turn endings and suchlike. Naturally, semantic and syntactic expectations provide quite considerable 'look-ahead' to humans, and in an ideal system these would be used in conjunction with acoustic analysis.

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 and Tsukahara [15], and is performed in several consecutive steps. Each step is described in detail below and the process flow is shown in Figure 1.

4.1. 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 in **/nailon/** 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 **/nailon/** 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.

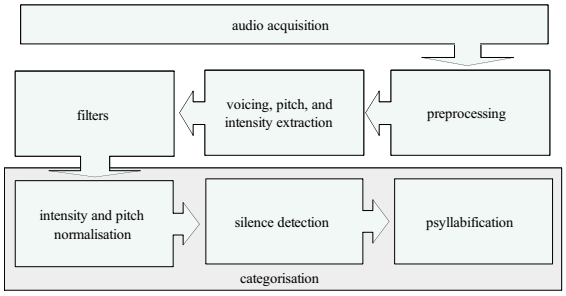

Figure 1. Process flow in ***/nailon/****.*

4.2. 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. Under its default configuration, however, **/nailon/** does not perform any preprocessing.

4.3. 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.

4.4. Filtering

Each time a get_f0 extraction results in a tick, a series of event driven processing steps take place. 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.

4.5. 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.

4.6. Silence detection

Many of the analyses we have used **/nailon/** for to date are refinements or additions of speech/silence decisions. For this reason **/nailon/** implements a simplistic speech activity detection (SAD). Note, however, that the **/nailon/** SAD is described here for completeness only; SAD is a well-researched area that lies outside our scope, and **/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.

4.7. 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, the intensity values of this record are searched backwards (last processing unit first) for a *convex hull* [loosely based on 16, 17] 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.

4.8. Classification

The normalised pitch, intensity, and voicing data extracted by **/nailon/** over a *psyllable* are intended for classification. In the implementation described below, F0 was categorised into HIGH, MID, or LOW depending on whether the pitch value is in the upper, mid or lower third of the range described by mean and standard deviation and a simplistic categorisation mechanism was used to classify each silence-preceding hull as a rise, fall, and or level prosodic pattern located high, mid or low in the speaker's F0 range.

5. /nailon/ applied to interaction control

In previous work [3], we explored to what extent the prosodic features extracted with **/nailon/** could be used to mimic the interaction control behaviour in conversations among humans. Specifically, we analysed one of the interlocutors in order to predict the interaction control decisions made by the other person taking part in the conversation. These predictions were evaluated with respect to whether there was a speaker change or not at that point in the conversation, that is, with respect to what the interlocutors actually did.

Each unit ending in a silent pause in the speech of the interlocutor being analysed was classified into one out of three categories: TURN-KEEPING, TURN-YIELDING, and DON'T KNOW. Units with low patterns were classified as suitable places for turn-taking (i.e. TURN-YIELDING); mid and level patterns were classified as unsuitable places (i.e. TURN-KEEPING); all other patterns, including high or rising, ended up in the garbage category DON'T KNOW. This tentative classification scheme was based on observations reported in the literature [e.g. 9, 12, 15], but it was in no way optimised or adapted to suit the speech material used.

This experiment showed that interaction control based on **/nailon/** features avoided 84% of the places where a system using silence duration thresholds only would have interrupted its users, while still recognizing 40% of the places where it was suitable to say something [cf. 3]. Interaction control decisions using prosodic information can furthermore be made considerably faster than in silence only systems. The decisions reported here were made after a 300-ms silence to be compared with silences ranging from 500 to 2000 ms in typical silence only systems [1].

6. 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. We 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. **/nailon/** can be used to improve the efficiency of identifying relevant places at which a machine can legitimately begin to talk to a human interlocutor by avoiding a substantial proportion of the places where it is unsuitable for the system to begin to talk, while still identifying roughly every second suitable place to talk. In addition, **/nailon/** can be used to improve the timing of contributions from the system by shortening system response times considerably.

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 [13]. Furthermore, machine learning should help improve results if used for training the interaction control decisions. We also intend to make future versions of **/nailon/** freely available.

In a long-term perspective, we would want to combine prosodic information with other sources of information, such as semantic completeness and visual interaction control cues, as well as to relate interaction control to other conversational phenomena such as grounding, error handling, and initiative.

7. Acknowledgements

This work was carried out within the CHIL project. CHIL (Computers in the Human Interaction Loop) is an Integrated Project under the European Commission's Sixth Framework Program (IP-506909).

8. References

[1] L. Ferrer, E. Shriberg, and A. Stolcke, "Is the speaker done yet? Faster and more accurate end-of-utterance detection using prosody in human-computer dialog," in *Proceedings of ICSLP 2002*, vol. 3. Denver, USA, 2002, pp. 2061-2064.

[2] E. Shriberg and A. Stolcke, "Direct Modeling of Prosody: An Overview of Applications in Automatic Speech Processing," in *Proceedings of Speech Prosody 2004* Nara, Japan, 2004, pp. 575-582.

[3] J. Edlund and M. Heldner, "Exploring Prosody in Interaction Control," *Phonetica*, vol. 62, pp. 215-226, 2005.

[4] L. Bell, J. Boye, and J. Gustafson, "Real-time handling of fragmented utterances," in *Proceedings of NAACL Workshop on Adaptation in Dialogue Systems*. Pittsburgh, PA, USA: Carnegie Mellon University, 2001, pp. 2-8.

[5] S. Duncan, Jr., "Some signals and rules for taking speaking turns in conversations," *Journal of Personality and Social Psychology*, vol. 23, pp. 283-292, 1972.

[6] C. E. Ford and S. A. Thompson, "Interactional units in conversation: syntactic, intonational, and pragmatic resources for the managment of turns," in *Interaction and grammar*, E. Ochs, E. A. Schegloff, and S. A. Thompson, Eds. Cambridge: Cambridge University Press, 1996, pp. 134-184.

[7] J. K. Local, J. Kelly, and W. H. G. Wells, "Towards a phonology of conversation: turn-taking in Tyneside English," *Journal of Linguistics*, vol. 22, pp. 411-437, 1986.

[8] G. Skantze and J. Edlund, "Robust interpretation in the Higgins spoken dialogue system," in *COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*. Norwich, UK, 2004.

[9] K. R. Thórisson, "Natural turn-taking needs no manual: Computational theory and model, from perception to action," in *Multimodality in language and speech systems*, B. Granström, D. House, and I. Karlsson, Eds. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2002, pp. 173-207.

[10] C. E. Ford and S. A. Thompson, "Interactional units in conversation: syntactic, intonational, and pragmatic resources for the management of turns," in *Interaction and grammar*, E. Ochs, E. A. Schegloff, and S. A. Thompson, Eds. Cambridge: Cambridge University Press, 1996, pp. 134-184.

[11] W. Wesseling and R. J. J. H. van Son, "Early preparation of experimentally elicited minimal responses," in *Proceedings of the Sixth SIGdial Workshop on Discourse and Dialogue*. Lisbon, Portugal: ISCA, 2005, pp. 11-18.

[12] J. Caspers, "Local speech melody as a limiting factor in the turn-taking system in Dutch," *Journal of Phonetics*, vol. 31, pp. 251-276, 2003.

[13] J. K. Local and J. Kelly, "Projection and 'silences': Notes on phonetic and conversational structure," *Human Studies*, vol. 9, pp. 185-204, 1986.

[14] M. Heldner and B. Megyesi, "Exploring the prosody-syntax interface in conversations," in *Proceedings ICPhS 2003*. Barcelona, 2003, pp. 2501-2504.

[15] N. Ward and W. Tsukahara, "Prosodic features which cue back-channel responses in English and Japanese," *Journal of Pragmatics*, vol. 32, pp. 1177-1207, 2000.

[16] A. W. Howitt, "Automatic Syllable Detection of Vowel Landmarks," MIT, 2000.

[17] P. Mermelstein, "Automatic segmentation of speech into syllabic units," *Journal of the Acoustical Society of America*, vol. 58, pp. 880-883, 1975.