The INPROTK 2012 Release: A Toolkit for Incremental Spoken Dialogue Processing

Timo Baumann[⋆] and *David Schlangen*[⋆⋆]

[⋆]Department of Informatics, University of Hamburg, 22527 Hamburg
[⋆⋆]Faculty of Linguistics and Literary Studies, Bielefeld University, 33501 Bielefeld
Email: baumann@informatik.uni-hamburg.de, david.schlangen@uni-bielefeld.de
Web: http://inprotk.sf.net, http://www.inpro.tk

Abstract

We describe the 2012 release of INPROTK[1], our "Incremental Processing Toolkit" which combines a powerful and extensible architecture for incremental processing with components for incremental speech recognition and, new to this release, incremental speech synthesis. These components work domain-independently; we also provide example implementations of higher-level components such as natural language understanding and dialogue management that are somewhat more tied to a particular domain. The toolkit is accompanied by evaluation tools for analysing timing behaviour, and we highlight some timing results on conversational speech input in this paper. We offer our toolkit to foster research in this new and exciting area, which promises to help increase the naturalness of behaviours that can be modelled in such systems.

1 Introduction

Recent work has shown that incremental (or *online*) processing of user input or generation of system output enables spoken dialogue systems to produce behaviour that is perceived as more natural than and preferable to that produced by systems that are bound by a turn-based processing mode [1–4]. There is still much left to find out about the best ways of modelling these behaviours in such systems, however. To foster research in this area, we are releasing a new version of our "Incremental Processing Toolkit" (INPROTK), which provides lower-level components (such as speech recognition and speech synthesis, but also a general modular processing architecture) and allows researchers to concentrate on higher-level modules (such as natural language understanding and dialogue modelling; for which we provide example implementations). We describe these components in the following, pointing out the differences and extensions to earlier releases [5]. Two competing incremental dialogue processing toolkits have been presented and compared to INPROTK previously [6].

[1]The toolkit is available as open-source at http://inprotk.sf.net; further information on the project is available at http://www.inpro.tk.

2 Incremental Processing Architecture

INPROTK realises the *IU*-model of incremental processing [7, 8], where incremental systems are conceptualised as consisting of a network of processing *modules*. Each module has a *left buffer*, a *processor*, and a *right buffer*, where the normal mode of processing is to take input from the left buffer, process it, and provide output in the right buffer, from where it goes to the next module's left buffer. (Top-down, expectation-based processing would work in the opposite direction.) As is shown in Figure 1, modules exchange *incremental units* (IUs), which are the smallest 'chunks' of information that can trigger connected modules into action. IUs typically are part of larger units; e.g., individual words as parts of an utterance, or frame elements as part of the representation of an utterance meaning. This relation of being part of the same larger unit is recorded through *same level* links; the units that were used in creating a given IU are linked via *grounded in* links. Thus, information is represented in the form of interconnected IUs which form a network representing the system's state.

Modules need to react to three basic situations: that IUs are *added* to a buffer, which triggers processing; that IUs that were erroneously hypothesised by an earlier module are *revoked*, which may trigger a revision of a module's own output; and that modules signal that they *commit* to an IU, that is, won't revoke it anymore (or, respectively, expect it to not be revoked anymore). INPROTK offers flexibility on how tightly or loosely modules are coupled in a system. It provides mechanisms for sending IU updates between processes via a light-weight remote procedure call protocol,[2] as well as for using shared memory within one process which allows each module to access the whole IU network via the links. INPROTK follows an event-based model, where modules create events that describe the edit they performed to the IU network, for which other modules can register as listeners. Additionally, modules can register listeners to specific IUs, e. g. to be notified when an output IU's delivery status changes. Individual modules and their interconnection to form a network are configured via a configuration file.

As opposed to our previous release [5], INPROTK

[2]In an earlier release, we have used the open agent architecture [9], but in the current branch this is not further developed.

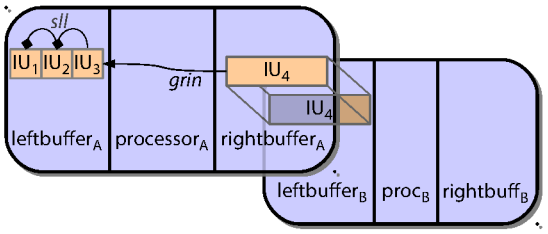

Figure 1: In the IU model, modules consist of a left buffer, a right buffer, and the processor proper. They are connected by super-imposing two buffers; IUs are linked by grounded-in (*grin* and same-level links (*sll*).

module communication is now completely encapsulated in the `IUModule` class. An implementing processor is called into action by a `leftBufferUpdate` method which gives access both to the edits to IUs in the left buffer since the last call, and to the list of IUs directly. The implementing processor must then notify its right buffer, either about the edits to the right buffer, or giving the content directly. Modules can be fully event-driven, only triggered into action by being notified of a hypothesis change, or they can run persistently, in order to create endogenous events like time-outs. Event-driven modules can run concurrently in separate threads or can be called sequentially by another module (which may seem to run counter the spirit of incremental processing, but can be advantageous for very quick computations for which the overhead of creating threads should be avoided). In the case of separate threads, which run at different update intervals, the left-buffer view will automatically be updated to its most recent state.

IUs are typed objects, where the base class `IU` specifies the links (same-level, grounded-in) that allow to create the IU network, handles update listeners, and the assignment of unique IDs. The payload and additional properties of an IU are specified for the IU's *type*. A design principle here is to make all relevant information available, while avoiding replication. For instance, an IU holding a bit of semantic representation can query which interval of input data it is based on, where this information is retrieved from the appropriate IUs by automatically following the grounded-in links. The lowest-level IUs are grounded in `BaseData`, which contains user-side input such as speech from the microphone, derived ASR feature vectors, camera feeds from a webcam, derived gaze information, etc., in several streams that can be accessed based on their timing information.

INPROTK comes with an extensive set of monitoring and profiling modules which can be linked into the module network at any point and allow to stream data to disk or to visualise it with TEDview [10]. INPROTK also supports several ways of simulating input (e. g. typed or read from a file) for debugging. All IU modules can also output log messages to the viewing tool directly (to ease graphic debugging of error cases in multi-threaded applications).

3 Incremental Speech Recognition

Our speech recognition module is based on the Sphinx-4 [11] toolkit and comes with acoustic models for German.[3] The module queries the ASR's current best hypothesis after each frame of audio and changes its output accordingly, adding or revoking `WordIUs` and notifying its listeners. Additionally, for each of the WordIUs, `SyllableIUs` and `SegmentIUs` are created and bound to the word (and to the syllable respectively) via the grounded-in hierarchy. Later modules in the pipeline are thus able to use this lower-level information (e.g. to disambiguate meaning based on prosodic aspects of words). For *prosodic processing*, we inject additional processors into Sphinx' acoustic frontend and provide pitch, loudness, and spectral tilt as *BaseData* features for further prosodic processing.

An ASR's current best hypothesis frequently changes during the recognition process with the majority of the changes not improving the result because wrong words are hypothesized intermittently [12, 13]. Every such change triggers all listening modules (and possibly their listeners), resulting in a lot of unnecessary processing. Furthermore, changes may actually deteriorate results, if a 'good' hypothesis is intermittently changed for worse. Therefore, we developed *hypothesis smoothing* approaches [12] which greatly reduce spurious edits in the output at the cost of some timeliness.

3.1 Evaluating Incremental Speech Recognition

While not part of the distribution proper, we think that it can only be useful for the field to agree on common evaluation metrics. Incremental processing brings new considerations of dynamics into the assessment of processing quality, and hence requires additional metrics compared to non-incremental processing. For example, raw WER is an unsuitable metric to evaluate incremental processing as it doesn't capture the dynamic evolution of incremental hypotheses over time.

We have refined and extended our previously outlined methodology and software for evaluating incremental speech recognition [12, 14]. An interactive tool now plots histograms showing when words have been first recognized *FO* (respectively finally decided on *FD*) relative to their position in the audio stream.

With the tool, smoothing parameters can be explored interactively to help finding the optimum for a given corpus. To test incremental performance on broad, conversational data, we trained and tested incremental ASR for Verbmobil-II. Results for *FD* are shown in Figure 2. Smoothing slightly increases mean *FD* times (from 160 ms to 218 ms after word end with smoothness of 12), while effectively reducing edit overhead (from 90 % of spurious dits down to 51 %).

[3] Other models compatible with Sphinx-4 can be used as well; our website explains the changes necessary for English.

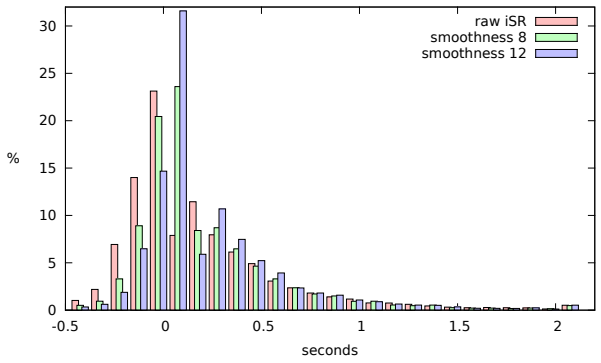

Figure 2: Final Decision (FD) for words in Verbmobil-II data, for raw iSR and two smoothing values, relative to the words' ends.

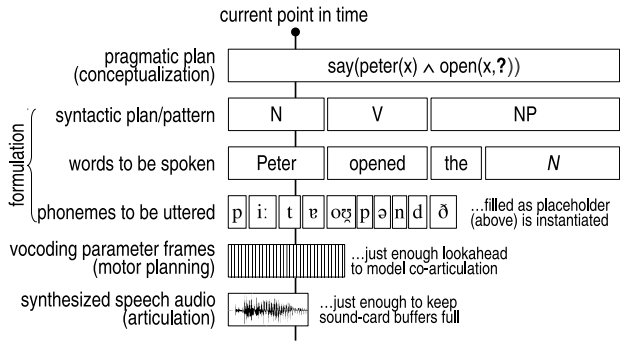

Figure 3: Hierarchic structure of incremental units describing an example utterance as it is being produced during delivery, showing the event-based just-in-time processing strategy.

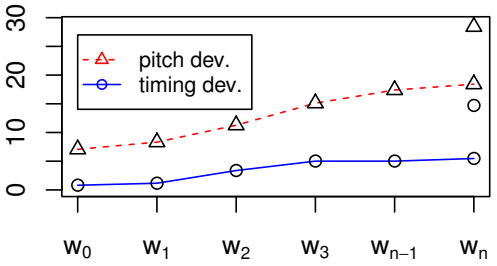

Figure 4: Deviation of pitch and timing plotted against lookahead. The more lookahead available, the better the results. (The single points on the right represent a trivial setting's performance.)

4 Incremental Speech Synthesis

Rounding out the toolkit is our new component for incremental speech synthesis (detailed in [15]) based on the non-incremental MaryTTS [16], with the following properties:

(a) It makes possible changes to the as-yet unspoken part of the ongoing utterance,
(b) as well as adaptations of delivery parameters such as speaking rate or pitch with very low latency.
(c) It autonomously makes delivery-related decisions (such as producing hesitations), and
(d) it makes available information about delivery status (what has been said; useful with barge-ins).
(e) And, last but not least, it runs in real time.

Figure 3 visualizes the internal data structures of the component, showing a triangular structure where on successive levels structure is built *just-in-time* (e.g., turning target phoneme sequences into vocoding parameters as in [17]) and hence can be changed with low cost, if necessary. We have evaluated the component in an application scenario [18] where it enabled highly favoured behaviour. We have also studied in detail the tradeoff of prosodic quality vs. timeliness of decisions [19]. Figure 4 plots the deviation of pitch and duration compared to non-incremental synthesis of the same utterance versus the amount of lookahead, that is, how far into the current phrase the next phrase becomes known. Unsurprisingly, results are better with more lookahead (further left in the figure, less timely behaviour). Slightly less than one phrase of lookahead leads to results that are similar (close to just-noticeable difference) to non-incremental synthesis.

5 Higher-level Components

As mentioned above, the more "higher-level" components in our toolkit are somewhat domain-specific (unlike the other components), and in any case are probably exactly those modules which users of the toolkit may want to substitute with their own.

5.1 Incremental NLU

We provide example implementations of a simple keyword-spotting 'NLU', as well as of statistically trained ones [20, 21]. A somewhat more traditional NLU component which could be more easily ported to other domains (by adapting lexicon and grammar) is described in [22]. It consists of a probabilistic, beam-search top-down parser (following [23]), which produces a principled semantic representation in the formalism *robust minimal recursion semantics* [24].

5.2 Incremental DM

An echo dialogue manager is available with INPROTK. Incremental dialogue management [25] is in its infancy, however it holds the promise of handling sub-utterance phenomena [26] like improved turn-taking [27].

5.3 Incremental NLG

While not part of INPROTK proper, we have recently plugged-in an incremental NLG component based on the SPUD microplanner [28]. Turning this component into an `IUmodule` to work together with incremental speech output was straightforward, exemplifying the coverage and ease-of-use of INPROTK.

6 Conclusions

We have sketched the major features of our "Incremental Processing Toolkit" INPROTK. While it is far from offering 'plug-and-play' ease of constructing incremental dialogue systems, we hope it will prove useful for other researchers insofar as it offers solutions to the more low-level problems that often are not one's main focus, but which need solving anyways to build incremental systems. We believe that INPROTK can be used to explore new interactional and conversational capabilities for spoken dialogue systems.

Acknowledgments

Much of the work decribed in this paper was funded by a grant from DFG in the Emmy Noether Programme.

References

[1] G. Aist, J. Allen, E. Campana, L. Galescu, C. Gomez Gallo, S. Stoness, M. Swift, and M. Tanenhaus, "Software architectures for incremental understanding of human speech," in *Proceedings of ICSLP*, (Pittsburgh, PA, USA), September 2006.

[2] G. Skantze and D. Schlangen, "Incremental dialogue processing in a micro-domain," in *Proceedings of EACL*, (Athens, Greece), pp. 745–753, March 2009.

[3] O. Buß, T. Baumann, and D. Schlangen, "Collaborating on utterances with a spoken dialogue system using an isu-based approach to incremental dialogue management," in *Proceedings of SigDial*, (Tokyo, Japan), pp. 233–236, September 2010.

[4] G. Skantze and A. Hjalmarsson, "Towards incremental speech generation in dialogue systems," in *Procs. of SigDial*, (Tokyo, Japan), pp. 1–8, September 2010.

[5] T. Baumann, O. Buß, and D. Schlangen, "InproTK in action: Open-source software for building german-speaking incremental spoken dialogue systems," in *Proceedings of ESSV*, (Berlin, Germany), 2010.

[6] D. Schlangen, T. Baumann, H. Buschmeier, O. Buß, S. Kopp, G. Skantze, and R. Yaghoubzadeh, "Middleware for incremental processing in conversational agents," in *Procs. of SigDial*, (Tokyo, Japan), 2010.

[7] D. Schlangen and G. Skantze, "A general, abstract model of incremental dialogue processing," in *Procs. of EACL*, (Athens, Greece), pp. 710–718, March 2009.

[8] D. Schlangen and G. Skantze, "A general, abstract model of incremental dialogue processing," *Dialogue and Discourse*, vol. 2, no. 1, pp. 83–111, 2011.

[9] A. Cheyer and D. Martin, "The open agent architecture," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4, pp. 143–148, March 2001.

[10] T. von der Malsburg, T. Baumann, and D. Schlangen, "Telida: A package for manipulation and visualisation of timed linguistic data," in *Proceedings of SigDial*, (London, UK), September 2009. Poster.

[11] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, "Sphinx-4: A flexible open source framework for speech recognition," Tech. Rep. SMLI TR2004-0811, Sun Microsystems Inc., 2004.

[12] T. Baumann, M. Atterer, and D. Schlangen, "Assessing and improving the performance of speech recognition for incremental systems," in *Proceedings of NAACL-HLT*, (Boulder, USA), May 2009.

[13] E. Selfridge, I. Arizmendi, P. Heeman, and J. Williams, "Stability and accuracy in incremental speech recognition," in *Proceedings of SigDial*, (Portland, USA), pp. 110–119, June 2011.

[14] T. Baumann, O. Buß, and D. Schlangen, "Evaluation and optimization of incremental processors," *Dialogue and Discourse*, vol. 2, no. 1, pp. 113–141, 2011.

[15] T. Baumann and D. Schlangen, "INPRO_iSS: A component for just-in-time incremental speech synthesis," in *Procs. of ACL System Demos*, (Jeju, Korea), 2012.

[16] M. Schröder and J. Trouvain, "The German text-to-speech synthesis system MARY: A tool for research, development and teaching," *International Journal of Speech Technology*, vol. 6, pp. 365–377, Oct. 2003.

[17] T. Dutoit, M. Astrinaki, O. Babacan, N. d'Alessandro, and B. Picart, "pHTS for Max/MSP: A streaming architecture for statistical parametric speech synthesis," Tech. Rep. 1, 3 2011.

[18] H. Buschmeier, T. Baumann, B. Dorsch, S. Kopp, and D. Schlangen, "Combining incremental language generation and incremental speech synthesis for adaptive information presentation," in *Proceedings of SigDial*, (Seoul, Korea), 2012.

[19] T. Baumann and D. Schlangen, "Evaluating prosodic processing for incremental speech synthesis," in *Proceedings of Interspeech*, (Portland, USA), 2012.

[20] D. Schlangen, T. Baumann, and M. Atterer, "Incremental reference resolution: The task, metrics for evaluation, and a bayesian filtering model that is sensitive to disfluencies," in *Proceedings of SigDial*, (London, UK), September 2009.

[21] S. Heintze, T. Baumann, and D. Schlangen, "Comparing local and sequential models for statistical incremental natural language understanding," in *Procs. of SigDial*, (Tokyo, Japan), pp. 9–16, September 2010.

[22] A. Peldszus, O. Buß, T. Baumann, and D. Schlangen, "Joint satisfaction of syntactic and pragmatic constraints improves incremental spoken language understanding," in *Proceedings of EACL*, (Avignon, France), April 2012.

[23] B. Roark, *Robust Probabilistic Predictive Syntactic Processing: Motivations, Models, and Applications*. PhD thesis, Department of Cognitive and Linguistic Sciences, Brown University, 2001.

[24] A. Copestake, "Robust minimal recursion semantics," tech. rep., Cambridge Computer Lab, 2006. Unpublished draft.

[25] O. Buß and D. Schlangen, "DIUM – an incremental dialogue manager that can produce self-corrections," in *Proceedings of SemDial 2011 (Los Angelogue)*, (Los Angeles, USA), 2011.

[26] O. Buß and D. Schlangen, "Modelling sub-utterance phenomena in spoken dialogue systems," in *Proceedings of SemDial (PozDial)*, (Poznan, Poland), 2010.

[27] E. Selfridge, I. Arizmendi, P. Heeman, and J. Williams, "A temporal simulator for developing turn-taking methods for spoken dialogue systems," in *Proceedings of SigDial*, (Seoul, Korea), 2012.

[28] M. Stone, C. Doran, B. Webber, T. Bleam, and M. Palmer, "Microplanning with communicative intentions: The SPUD system," *Computational Intelligence*, vol. 19, pp. 311–381, 2003.