The INPROTK 2012 Release

Timo Baumann
Department for Informatics
University of Hamburg, Germany
baumann@informatik.uni-hamburg.de

David Schlangen
Faculty of Linguistics and Literary Studies
Bielefeld University, Germany
david.schlangen@uni-bielefeld.de

Abstract

We describe the 2012 release of our "Incremental Processing Toolkit" (INPROTK)[1], 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 fairly 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. We offer this release of the 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

As recent work has shown, 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 (Aist et al., 2006; Skantze and Schlangen, 2009; Buß et al., 2010; Skantze and Hjalmarsson, 2010). 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).[2] We describe these components in the following, pointing out the differences and extensions to earlier releases (Baumann et al., 2010).

2 An Incremental Processing Architecture

INPROTK realises the *IU*-model of incremental processing (Schlangen and Skantze, 2009; Schlangen and Skantze, 2011), 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.) 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 to it via *grounded in* links. Modules have to be able 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

[1] The code of the toolkit and some example applications have been released as open-source at `http://inprotk.sourceforge.net`.

[2] An alternative to the toolkit described here is *jindigo* (Skantze and Hjalmarsson, 2010), `http://www.jindigo.net`.

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,[3] as well as for using shared memory within one (Java) process. INPROTK follows an event-based model, where modules create events, for which other modules can register as listeners. Module networks are configured via a system configuration file which specifies which modules *listen* to which.

As opposed to our previous release (Baumann et al., 2010), INPROTK module communication is now completely encapsulated in the `IUModule` class. An implementing processor is called into action by a 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.

INPROTK also 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 online through a viewing tool (von der Malsburg et al., 2009), as well as different ways to simulate input (e.g., typed or read from a file) for debugging. All IUmodules can also output loggging messages to the viewing tool directly (to ease graphic debugging of error cases in multi-threaded applications).

[3] In an earlier release, we used OAA (Cheyer and Martin, 2001), which however turned out to be too slow.

3 Incremental Speech Recognition

Our speech recognition module is based on the Sphinx-4 (Walker et al., 2004) toolkit and comes with acoustic models for German.[4] The module queries the ASR's current best hypothesis after each frame of audio and changes its output accordingly, adding or revoking `WordIU`s and notifying its listeners. Additionally, for each of the WordIUs, `SyllableIU`s and `SegmentIU`s 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 which provide features for further prosodic processing (pitch, loudness, and spectral tilt). In this way, IUs are able to access the precise acoustic data (in raw and processed forms).

An ASR's current best hypothesis frequently changes during the recognition process with the majority of the changes not improving the result. 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 (Baumann et al., 2009) which greatly reduce spurious edits in the output at the cost of some timeliness: With a lag of 320 ms we reduced the amount of spurious edits to 10 % from an initial 90 %. The current implementation of hypothesis smoothing is taylored specifically towards ASR output, but other input modules (like gesture or facial expression recognition) could easily be smoothed with similar methods.

4 Incremental NLU and DM

As mentioned above, the more "higher-level" components in our toolkit are more domain-specific than the other components, and in any case are probably exactly those modules which users of the toolkit may want to substitute with their own. Nevertheless, we provide example implementations of a simple keyword-spotting 'NLU', as well as statistically

[4] Models for English, French and other languages are available from the Sphinx' distribution and from `http://www.voxforge.org`.

trained ones (Schlangen et al., 2009; Heintze et al., 2010).

We have recently built a somewhat more traditional NLU component which could be more easily ported to other domains (by adapting lexicon and grammar). It consists of a probabilistic, beam-search top-down parser (following (Roark, 2001)), which produces a principled semantic representation in the formalism *robust minimal recursion semantics* (Copestake, 2006). This component is described in more detail in (Peldszus et al., 2012).

5 Incremental Speech Synthesis

Rounding out the toolkit is our new component for incremental speech synthesis, which has the following properties:

(a) It makes possible changes to the as-yet unspoken part of the ongoing utterance,
(b) allows 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 provides information about delivery status (e. g. useful in case of barge-ins).
(e) And, last but not least, it runs in real time.

Figure 1 provides a look into 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) and hence can be changed with low cost, if necessary. We have evaluated the component in an application scenario where it proved to increase perceived naturalness, and have also studied the tradeoff between look-ahead and prosodic quality. To this end, Figure 2 plots the deviation of the prosodic parameters *pitch* and *timing* from that of a non-incremental synthesis of the same utterance versus the amount of *look-ahead*, that is, how far into the current phrase the next phrase becomes known. It shows that best results are achieved if the next phrase that is to be synthesized becomes known no later than one or two words into the current phrase (w_0 or w_1).

6 Evaluation of Incremental Processors

While not part of the toolkit proper, we think that it can only be useful for the field to agree on common evaluation metrics. Incremental processing brings

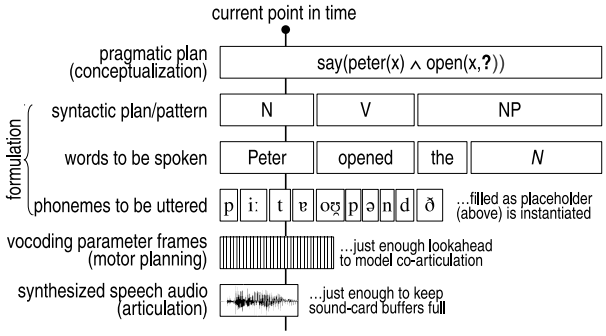

Figure 1: Hierarchic structure of incremental units describing an example utterance as it is being produced during delivery, showing the event-based just-in-time processing strategy.

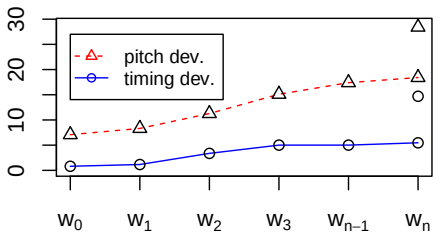

Figure 2: Deviation of pitch and timing plotted against lookahead (right context available for incremental synthesis). The more lookahead available, the better the results.

new considerations of dynamics into the assessment of processing quality, and hence requires additional metrics compared to non-incremental processing. In (Baumann et al., 2011) we have proposed a family of such metrics, and we provide an evaluation framework for analysing incremental ASR performance as part of our distribution.

7 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 before more interesting things can be done. We look forward to what these interesting things may be that others will build.

Acknowledgments

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

References

G.S. Aist, J. Allen, E. Campana, L. Galescu, C.A. Gomez Gallo, S. Stoness, M. Swift, and M Tanenhaus. 2006. Software architectures for incremental understanding of human speech. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Pittsburgh, PA, USA, September.

Timo Baumann, Michaela Atterer, and David Schlangen. 2009. Assessing and improving the performance of speech recognition for incremental systems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT) 2009 Conference*, Boulder, Colorado, USA, May.

Timo Baumann, Okko Buß, and David Schlangen. 2010. InproTK in action: Open-source software for building german-speaking incremental spoken dialogue systems. In *Proceedings of ESSV 2010*, Berlin, Germany.

Timo Baumann, Okko Buß, and David Schlangen. 2011. Evaluation and optimization of incremental processors. *Dialogue and Discourse*, 2(1):113–141.

Okko Buß, Timo Baumann, and David Schlangen. 2010. Collaborating on utterances with a spoken dialogue system using an isu-based approach to incremental dialogue management. In *Proceedings of the SIGdial 2010 Conference*, pages 233–236, Tokyo, Japan, September.

Adam Cheyer and David Martin. 2001. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1):143–148, March. OAA.

Ann Copestake. 2006. Robust minimal recursion semantics. Technical report, Cambridge Computer Lab. Unpublished draft.

Silvan Heintze, Timo Baumann, and David Schlangen. 2010. Comparing local and sequential models for statistical incremental natural language understanding. In *Proceedings of the SIGdial 2010 Conference*, pages 9–16, Tokyo, Japan, September.

Andreas Peldszus, Okko Buß, Timo Baumann, and David Schlangen. 2012. Joint satisfaction of syntactic and pragmatic constraints improves incremental spoken language understanding. In *Proceedings of the Conference of the European Association for Computational Linguistics (EACL 2012)*, Avignon, France, April.

Brian Roark. 2001. *Robust Probabilistic Predictive Syntactic Processing: Motivations, Models, and Applications*. Ph.D. thesis, Department of Cognitive and Linguistic Sciences, Brown University.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pages 710–718, Athens, Greece, March.

David Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue and Discourse*, 2(1):83–111.

David Schlangen, Timo Baumann, and Michaela Atterer. 2009. Incremental reference resolution: The task, metrics for evaluation, and a bayesian filtering model that is sensitive to disfluencies. In *Proceedings of SIGdial 2009, the 10th Annual SIGDIAL Meeting on Discourse and Dialogue*, London, UK, September.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards incremental speech generation in dialogue systems. In *Proceedings of the SIGdial 2010 Conference*, pages 1–8, Tokyo, Japan, September.

Gabriel Skantze and David Schlangen. 2009. Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pages 745–753, Athens, Greece, March.

Titus von der Malsburg, Timo Baumann, and David Schlangen. 2009. Telida: A package for manipulation and visualisation of timed linguistic data. In *Proceedings of the Poster Session at SIGdial 2009, the 10th Annual SIGDIAL Meeting on Discourse and Dialogue*, London, UK, September.

Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. 2004. Sphinx-4: A flexible open source framework for speech recognition. Technical Report SMLI TR2004-0811, Sun Microsystems Inc.