Incremental Dialogue Processing in a Micro-Domain

Gabriel Skantze[1]
Dept. of Speech, Music and Hearing
KTH, Stockholm, Sweden
gabriel@speech.kth.se

David Schlangen
Department of Linguistics
University of Potsdam, Germany
das@ling.uni-potsdam.de

Abstract

This paper describes a fully incremental dialogue system that can engage in dialogues in a simple domain, number dictation. Because it uses incremental speech recognition and prosodic analysis, the system can give rapid feedback as the user is speaking, with a very short latency of around 200ms. Because it uses incremental speech synthesis and self-monitoring, the system can react to feedback from the user as the system is speaking. A comparative evaluation shows that naïve users preferred this system over a non-incremental version, and that it was perceived as more human-like.

1 Introduction

A traditional simplifying assumption for spoken dialogue systems is that the dialogue proceeds with strict turn-taking between user and system. The minimal unit of processing in such systems is the *utterance*, which is processed in whole by each module of the system before it is handed on to the next. When the system is speaking an utterance, it assumes that the user will wait for it to end before responding. (Some systems accept barge-ins, but then treat the interrupted utterance as basically unsaid.)

Obviously, this is not how natural human-human dialogue proceeds. Humans understand and produce language *incrementally* – they use multiple knowledge sources to determine when it is appropriate to speak, they give and receive backchannels in the middle of utterances, they start to speak before knowing exactly what to say, and they incrementally monitor the listener's reactions to what they say (Clark, 1996).

This paper presents a dialogue system, called NUMBERS, in which all components operate incrementally. We had two aims: First, to explore technical questions such as how the components of a modularized dialogue system should be arranged and made to interoperate to support incremental processing, and which requirements incremental processing puts on dialogue system components (e.g., speech recognition, prosodic analysis, parsing, discourse modelling, action selection and speech synthesis). Second, to investigate whether incremental processing can help us to better model certain aspects of human behaviour in dialogue systems – especially turn-taking and feedback – and whether this improves the user's experience of using such a system.

2 Incremental dialogue processing

All dialogue systems are 'incremental', in some sense – they proceed in steps through the exchange of 'utterances'. However, incremental processing typically means more than this; a common requirement is that processing starts before the input is complete and that the first output increments are produced as soon as possible (e.g., Kilger & Finkler, 1995). Incremental modules hence are those where "Each processing component will be triggered into activity by a minimal amount of its characteristic input" (Levelt, 1989). If we assume that the 'characteristic input' of a dialogue system is the utterance, this principle demands that 'minimal amounts' of an utterance already trigger activity. It should be noted though, that there is a trade-off between responsiveness and output quality, and that an incremental process therefore should produce output only as soon as it is possible to reach a desired output quality criterion.

2.1 Motivations & related work

The claim that humans do not understand and produce speech in utterance-sized chunks, but

[1] The work reported in this paper was done while the first author was at the University of Potsdam.

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 745–753,
Athens, Greece, 30 March – 3 April 2009.

rather incrementally, can be supported by an impressive amount of psycholinguistic literature on the subject (e.g., Tanenhaus & Brown-Schmidt, 2008; Levelt, 1989). However, when it comes to spoken dialogue systems, the dominant minimal unit of processing has been the utterance. Moreover, traditional systems follow a very strict sequential processing order of utterances – interpretation, dialogue management, generation – and there is most often no monitoring of whether (parts of) the generated message is successfully delivered.

Allen et al. (2001) discuss some of the shortcomings of these assumptions when modelling more conversational human-like dialogue. First, they fail to account for the frequently found mid-utterance reactions and feedback (in the form of acknowledgements, repetition of fragments or clarification requests). Second, people often seem to start to speak before knowing exactly what to say next (possibly to grab the turn), thus producing the utterance incrementally. Third, when a speaker is interrupted or receives feedback in the middle of an utterance, he is able to continue the utterance from the point where he was interrupted.

Since a non-incremental system needs to process the whole user utterance using one module at a time, it cannot utilise any higher level information for deciding when the user's turn or utterance is finished, and typically has to rely only on silence detection and a time-out. Silence, however, is not a good indicator: sometimes there is silence but no turn-change is intended (e.g., hesitations), sometimes there isn't silence, but the turn changes (Sacks et al., 1974). Speakers appear to use other knowledge sources, such as prosody, syntax and semantics to detect or even project the end of the utterance. Attempts have been made to incorporate such knowledge sources for turn-taking decisions in spoken dialogue systems (e.g., Ferrer et al., 2002; Raux & Eskenazi, 2008). To do so, incremental dialogue processing is clearly needed.

Incremental processing can also lead to better use of resources, since later modules can start to work on partial results and do not have to wait until earlier modules have completed processing the whole utterance. For example, while the speech recogniser starts to identify words, the parser can already add these to the chart. Later modules can also assist in the processing and for example resolve ambiguities as they come up. Stoness et al. (2004) shows how a reference resolution module can help an incremental parser with NP suitability judgements. Similarly, Aist et al. (2006) shows how a VP advisor could help an incremental parser.

On the output side, an incremental dialogue system could *monitor* what is actually happening to the utterance it produces. As discussed by Raux & Eskenazi (2007), most dialogue managers operate asynchronously from the output components, which may lead to problems if the dialogue manager produces several actions and the user responds to one of them. If the input components do not have any information about the timing of the system output, they cannot relate them to the user's response. This is even more problematic if the user reacts (for example with a backchannel) in the middle of system utterances. The system must then relate the user's response to the parts of its planned output it has managed to realise, but also be able to stop speaking and possibly continue the interrupted utterance appropriately. A solution for handling mid-utterance responses from the user is proposed by Dohsaka & Shimazu (1997). For incremental generation and synthesis, the output components must also cope with the problem of revision (discussed in more detail below), which may for example lead to the need for the generation of speech repairs, as discussed by Kilger & Finkler (1995).

As the survey above shows, a number of studies have been done on incrementality in different areas of language processing. There are, however, to our knowledge no studies on how the various components could or should be integrated into a complete, fully incremental dialogue system, and how such a system might be perceived by naïve users, compared to a non-incremental system. This we provide here.

2.2 A general, abstract model

The NUMBERS system presented in this paper can be seen as a specific instance (with some simplifying assumptions) of a more general, abstract model that we have developed (Schlangen & Skantze, 2009). We will here only briefly describe the parts of the general model that are relevant for the exposition of our system.

We model the dialogue processing system as a collection of connected processing *modules*. The smallest unit of information that is communicated along the connections is called the *incremental unit* (IU), the unit of the "minimal amount of characteristic input". Depending on what the module does, IUs may be audio frames, words, syntactic phrases, communicative acts,

etc. The processing module itself is modelled as consisting of a *Left Buffer* (LB), the *Processor* proper, and a *Right Buffer* (RB). An example of two connected modules is shown in Figure 1. As IU_1 enters the LB of module A, it may be *consumed* by the processor. The processor may then produce new IUs, which are posted on the RB (IU_2 in the example). As the example shows, the modules in the system are connected so that an IU posted on the RB in one module may be consumed in the LB of another module. One RB may of course be connected to many other LB's, and vice versa, allowing a range of different network topologies.

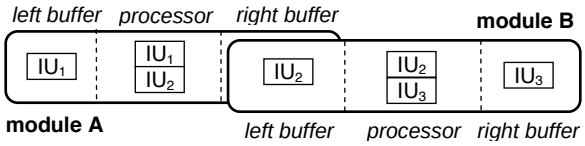

Figure 1: Two connected modules.

In the NUMBERS system, information is only allowed to flow from left to right, which means that the LB may be regarded as the input buffer and the RB as the output buffer. However, in the general model, information may flow in both directions.

A more concrete example is shown in Figure 2, which illustrates a module that does incremental speech recognition. The IUs consumed from the LB are audio frames, and the IUs posted in the RB are the words that are recognised.

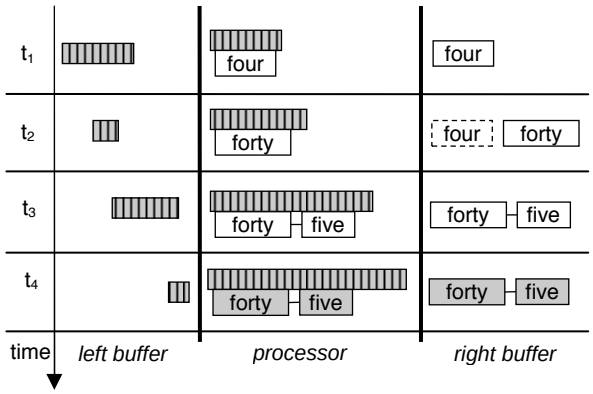

Figure 2: Speech recognition as an example of incremental processing.

We identify three different generic module operations on IUs: *update*, *purge* and *commit*. First, as an IU is added to the LB, the processor needs to **update** its internal state. In the example above, the speech recogniser has to continuously add incoming audio frames to its internal state, and as soon as the recogniser receives enough audio frames to decide that the word "four" is a good-enough candidate, the IU holding this word will be put on the RB (time-point t_1). If a processor only expects IUs that extend the rightmost IU currently produced, we can follow Wirén (1992) in saying that it is only *left-to-right incremental*. A *fully incremental* system (which we aim at here), on the other hand, also allows insertions and/or revisions.

An example of *revision* is illustrated at time-point t_2 in Figure 2. As more audio frames are consumed by the recogniser, the word "four" is no longer the best candidate for this stretch of audio. Thus, the module must now revoke the IU holding the word "four" (marked with a dotted outline) and add a new IU for the word "forty". All other modules consuming these IUs must now **purge** them from their own states and possibly revoke other IUs. By allowing revision, a module may produce tentative results and thus make the system more responsive.

As more audio frames are consumed in the example above, a new word "five" is identified and added to the RB (time-point t_3). At time-point t_4, no more words are identified, and the module may decide to **commit** to the IUs that it has produced (marked with a darker shade). A committed IU is guaranteed to not being revoked later, and can hence potentially be removed from the processing window of later modules, freeing up resources.

3 Number dictation: a micro-domain

Building a fully incremental system with a behaviour more closely resembling that of human dialogue participants raises a series of new challenges. Therefore, in order to make the task more feasible, we have chosen a very limited domain – what might be called a *micro-domain* (cf. Edlund et al., 2008): the dictation of number sequences. In this scenario, the user dictates a sequence of numbers (such as a telephone number or a credit card number) to the dialogue system. This is a very common situation in commercial telephone-based dialogue systems, which however operate in a non-incremental manner: The user is first asked to read out the whole number sequence, which the system then confirms. Should the recognition be incorrect, the user has to repeat the whole sequence again. In an incremental version of this scenario, the system might give continuous feedback (such as acknowledgements and clarification requests) as the user is reading the

sequence. When the system repeats the sequence, the user is also given the opportunity to give feedback and make corrections at any time. Obviously, such a dialogue will result in many more speaker-shifts, which in turn increases the need for the rapid turn-taking. Thus, while the number dictation task might appear simple at a first glance (very limited vocabulary, simple syntax and semantics), there are several issues – not typically considered in most dialogue systems – which an incremental and more human-like dialogue system must address.

To explore dialogue phenomena in this domain, we gave 8 pairs of subjects the task of dictating number sequences to each other. The subjects were sitting in different rooms and used headsets to communicate. To induce more feedback behaviour, noise was randomly inserted in the utterances from the information giver (as in the setting used by Schlangen & Fernández (2007)). The subjects were all speaking German (their native tongue). The recorded dialogues were then transcribed and annotated. The observed dialogue strategies directly influenced the design of our system (see below). Table 1 shows an example of a dialogue between a user and the system and illustrates some of the phenomena we intended to model.

U.01	two five three -
S.02	*okay*
U.03	one forty, five -
S.04	*sorry, do you mean one and then forty and then five?*
U.05	no, one forty five
S.06	*okay, one forty five*
U.07	yeah, and six eight four -
S.08	*okay*
U.09	seven twelve nine.
S.10	*so, from the start, two five three -*
U.11	yeah
S.12	*one forty five -*
S.13	*six six four -*
U.14	no, six eight four
S.15	*okay, six eight four -*
S.16	*and then seven twelve nine.*
U.17	that's right

Table 1: An example dialogue between a user and the NUMBERS system.

Just like in this example, a common strategy for the subjects was to first let the information giver read the number sequence and then switch roles and let the information follower repeat it. Instead of reading out the whole number sequence at once, subjects tended to package pieces of information into what Clark (1996) refers to as *installments* (in this case small groups of numbers). After each installment, the other speaker may react by giving an acknowledgement (as in S.02) a clarification request (as in S.04), a correction (as in U.14), or do nothing (as after S.12).

As there are a lot of speaker shifts, there needs to be a mechanism for rapid turn taking. In the example above, the system must recognize that the last digit in U.01, U.03, U.05 and U.07 ends an installment and calls for a reaction, while the last digit in U.09 ends the whole sequence. One information source that has been observed to be useful for this is prosody (Koiso et al., 1998). When analysing the recorded dialogues, it seemed like mid-sequence installments most often ended with a prolonged duration and a rising pitch, while end-sequence installments most often ended with a shorter duration and a falling pitch. How prosody is used by the NUMBERS system for this classification is described in section 4.2.

4 The NUMBERS system components

The NUMBERS system has been implemented using the HIGGINS spoken dialogue system framework (Skantze, 2007). All modules have been adapted and extended to allow incremental processing. It took us roughly 6 months to implement the changes described here to a fully working baseline system. Figure 3 shows the architecture of the system[2].

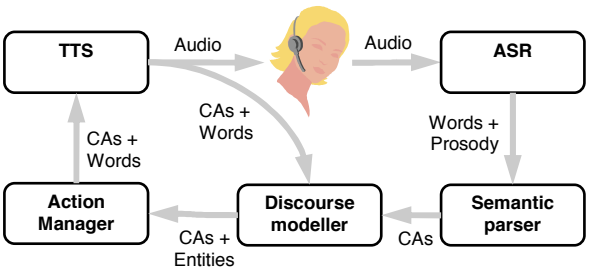

Figure 3: The system architecture.
CA = communicative act.

This is pretty much a standard dialogue system layout, with some exceptions that will be discussed below. Most notably perhaps is that dialogue management is divided into a discourse modelling module and an action manager. As can

[2] A video showing an example run of the system has been uploaded to http://www.youtube.com/watch?v=_rDkb1K1si8

be seen in the figure, the discourse modeller also receives information about what the system itself says. The modules run asynchronously in separate processes and communicate by sending XML messages containing the IUs over sockets.

We will now characterize each system module by what kind of IUs they consume and produce, as well as the criteria for committing to an IU.

4.1 Speech recognition

The automatic speech recognition module (ASR) is based on the Sphinx 4 system (Lamere et al., 2003). The Sphinx system is capable of incremental processing, but we have added support for producing incremental results that are compatible with the HIGGINS framework. We have also added prosodic analysis to the system, as described in 4.2. For the NUMBERS domain, we use a very limited context-free grammar accepting number words as well as some expressions for feedback and meta-communication.

An illustration of the module buffers is shown in Figure 2 above. The module consumes audio frames (each 100 msec) from the LB and produces words with prosodic features in the RB. The RB is updated every time the sequence of top word hypotheses in the processing windows changes. After 2 seconds of silence has been detected, the words produced so far are committed and the speech recognition search space is cleared. Note that this does not mean that other components have to wait for this amount of silence to pass before starting to process or that the system cannot respond until then – incremental results are produced as soon as the ASR determines that a word has ended.

4.2 Prosodic analysis

We implemented a simple form of prosodic analysis as a data processor in the Sphinx frontend. Incremental F0-extraction is done by first finding pitch candidates (on the semitone scale) for each audio frame using the SMDSF algorithm (Liu et al., 2005). An optimal path between the candidates is searched for, using dynamic programming (maximising candidate confidence scores and minimising F0 shifts). After this, median smoothing is applied, using a window of 5 audio frames.

In order for this sequence of F0 values to be useful, it needs to be parameterized. To find out whether pitch and duration could be used for the distinction between mid-sequence installments and end-sequence installments, we did a machine learning experiment on the installment-ending digits in our collected data. There were roughly an equal amount of both types, giving a majority class baseline of 50.9%.

As features we calculated a delta pitch parameter for each word by computing the sum of all F0 shifts (negative or positive) in the pitch sequence. (Shifts larger than a certain threshold (100 cents) were excluded from the summarization, in order to sort out artefacts.) A duration parameter was derived by calculating the sum of the phoneme lengths in the word, divided by the sum of the average lengths of these phonemes in the whole data set. Both of these parameters were tested as predictors separately and in combination, using the Weka Data Mining Software (Witten & Frank, 2005). The best results were obtained with a J.48 decision tree, and are shown in Table 2.

Baseline	50.9%
Pitch	81.2%
Duration	62.4%
Duration + Pitch	80.8%

Table 2: The results of the installment classification (accuracy).

As the table shows, the best predictor was simply to compare the delta pitch parameter against an optimal threshold. While the performance of 80.8% is significantly above baseline, it could certainly be better. We do not know yet whether the sub-optimal performance is due to the fact that the speakers did not always use these prosodic cues, or whether there is room for improvement in the pitch extraction and parameterization.

Every time the RB of the ASR is updated, the delta pitch parameter is computed for each word and the derived threshold is used to determine a pitch slope class (rising/falling) for the word. (Note that there is no class for a flat pitch. This class is not really needed here, since the digits within installments are followed by no or only very short pauses.) The strategy followed by the system then is this: when a digit with a rising pitch is detected, the system *plans* to immediately give a mid-sequence reaction utterance, and does so if indeed no more words are received. If a digit with a falling pitch is detected, the system plans an end-of-sequence utterance, but waits a little bit longer before producing it, to see if there really are no more words coming in. In other words, the system bases its turn-taking decisions on a combination of ASR, prosody and silence-thresholds, where the length of the threshold

differs for different prosodic signals, and where reactions are planned already during the silence. (This is in contrast to Raux & Eskenazi (2008), where context-dependent thresholds are used as well, but only simple end-pointing is performed.)

The use of prosodic analysis in combination with incremental processing allows the NUMBERS system to give feedback after mid-sequence installments in about 200 ms. This should be compared with most dialogue systems which first use a silence threshold of about 750-1500 msec, after which each module must process the utterance.

4.3 Semantic parsing

For semantic parsing, the incremental processing in the HIGGINS module PICKERING (Skantze & Edlund, 2004) has been extended. PICKERING is based on a modified chart parser which adds automatic relaxations to the CFG rules for robustness, and produces semantic interpretations in the form of concept trees. It can also use features that are attached to incoming words, such as prosody and timestamps. For example, the number groups in U.03 and U.05 in Table 1 render different parses due to the pause lengths between the words.

The task of PICKERING in the NUMBERS domain is very limited. Essentially, it identifies communicative acts (CAs), such as number installments. The only slightly more complex parsing is that of larger numbers such as "twenty four". There are also cases of "syntactic ambiguity", as illustrated in U.03 in the dialogue example above ("forty five" as "45" or "40 5"). In the NUMBERS system, only 1-best hypotheses are communicated between the modules, but PICKERING can still assign a lower parsing confidence score to an ambiguous interpretation, which triggers a clarification request in S.04.

Figure 4 show a very simple example of the incremental processing in PICKERING. The LB contains words with prosodic features produced by the ASR (compare with Figure 2 above). The RB consists of the CAs that are identified. Each time a word is added to the chart, PICKERING continues to build the chart and then searches for an optimal sequence of CAs in the chart, allowing non-matching words in between. To handle revision, a copy of the chart is saved after each word has been added.

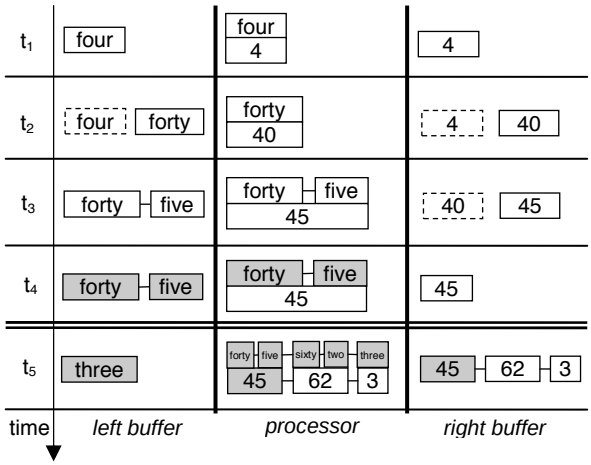

Figure 4: Incremental parsing. There is a jump in time between *t4* and *t5*.

As can be seen at time-point t_4, even if all words that a CA is based on are committed, the parser does not automatically commit the CA. This is because later words may still cause a revision of the complex output IU that has been built. As a heuristic, PICKERING instead waits until a CA is followed by three words that are not part of it until it commits, as shown at time-point t_5. After a CA has been committed, the words involved may be cleared from the chart. This way, PICKERING parses a "moving window" of words.

4.4 Discourse modelling

For discourse modelling, the HIGGINS module GALATEA (Skantze, 2008) has been extended to operate incrementally. The task of GALATEA is to interpret utterances in their context by transforming ellipses into full propositions, indentify discourse entities, resolve anaphora and keep track of the grounding status of concepts (their confidence score and when they have been grounded in the discourse). As can be seen in Figure 3, GALATEA models both utterances from the user as well as the system. This makes it possible for the system to monitor its own utterances and relate them to the user's utterances, by using timestamps produced by the ASR and the speech synthesiser.

In the LB GALATEA consumes CAs from both the user (partially committed, as seen in Figure 4) and the system (always committed, see 4.6). In the RB GALATEA produces an incremental discourse model. This model contains a list of resolved communicative acts and list of resolved discourse entities. This model is then consulted by an action manager which decides what the system should do next. The discourse model is

committed up to the point of the earliest non-committed incoming CA. In the NUMBERS domain, the discourse entities are the number installments.

4.5 Action management

Based on the discourse model (from the LB), the action manager (AM) generates system actions (CAs) in semantic form (for GALATEA) with an attached surface form (for the TTS), and puts them on the RB. (In future extensions of the system, we will add an additional generation module that generates the surface form from the semantic form.) In the NUMBERS system, possible system actions are acknowledgements, clarification requests and repetitions of the number sequence. The choice of actions to perform is based on the grounding status of the concepts (which is represented in the discourse model). For example, if the system has already clarified the first part of the number sequence due to an ambiguity, it does not need to repeat this part of the sequence again.

The AM also attaches a desired timing to the produced CA, relative to the end time of last user utterance. For example, if a number group with a final rising pitch is detected, the AM may tell the TTS to execute the CA immediately after the user has stopped speaking. If there is a falling pitch, it may tell the TTS to wait until 500 msec of silence has been detected from the user before executing the action. If the discourse model gets updated during this time, the AM may revoke previous CAs and replace them with new ones.

4.6 Speech synthesis

A diphone MBROLA text-to-speech synthesiser (TTS) is used in the system (Dutoit et al., 1996), and a wrapper for handling incremental processing has been implemented. The TTS consumes words linked to CAs from the LB, as produced by the AM. As described above, each CA has a timestamp. The TTS places them on a queue, and prepares to synthesise and start sending the audio to the speakers. When the system utterance has been played, the corresponding semantic concepts for the CA are sent to GALATEA. If the TTS is interrupted, the semantic fragments of the CA that corresponds to the words that were spoken are sent. This way, GALATEA can monitor what the system actually says and provide the AM with this information. Since the TTS only sends (parts of) the CAs that have actually been spoken, these are always marked as committed.

There is a direct link from the ASR to the TTS as well (not shown in Figure 3), informing the TTS of start-of-speech and end-of-speech events. As soon as a start-of-speech event is detected, the TTS stops speaking. If the TTS does not receive any new CAs from the AM as a consequence of what the user said, it automatically resumes from the point of interruption. (This implements a "reactive behaviour" in the sense of (Brooks, 1991), which is outside of the control of the AM.)

An example of this is shown in Table 1. After U.09, the AM decides to repeat the whole number sequence and sends a series of CAs to the TTS for doing this. After S.10, the user gives feedback in the form of an acknowledgement (U.11). This causes the TTS to make a pause. When GALATEA receives the user feedback, it uses the time-stamps to find out that the feedback is related to the number group in S.10 and the grounding status for this group is boosted. When the AM receives the updated discourse model, it decides that this does not call for any revision to the already planned series of actions. Since the TTS does not receive any revisions, it resumes the repetition of the number sequence in S.12.

The TTS module is fully incremental in that it can stop and resume speaking in the middle of an utterance, revise planned output, and can inform other components of what (parts of utterances) has been spoken. However, the actual text-to-speech processing is done before the utterance starts and not yet incrementally as the utterance is spoken, which could further improve the efficiency of the system. This is a topic for future research, together with the generation of hidden and overt repair as discussed by Kilger & Finkler (1995).

5 Evaluation

It is difficult to evaluate complete dialogue systems such as the one presented here, since there are so many different components involved (but see Möller et al. (2007) for methods used). In our case, we're interested in the benefits of a specific aspect, though, namely incrementality. No evaluation is needed to confirm that an incremental system such as this allows more flexible turn-taking and that it can potentially respond faster – this is so by design. However, we also want this behaviour to result in an improved user experience. To test whether we have achieved this, we implemented for comparison a non-incremental version of the system, very much like a standard number dictation dialogue in a commercial application. In this version, the user

is asked to read out the whole number sequence in one go. After a certain amount of silence, the system confirms the whole sequence and asks a yes/no question whether it was correct. If not, the user has to repeat the whole sequence.

Eight subjects were given the task of using the two versions of the system to dictate number sequences (in English) to the system. (The subjects were native speakers of German with a good command of English.) Half of the subjects used the incremental version first and the other half started with the non-incremental version. They were asked to dictate eight number sequences to each version, resulting in 128 dialogues. For each sequence, they were given a time limit of 1 minute. After each sequence, they were asked whether they had succeeded in dictating the sequence or not, as well as to mark their agreement (on a scale from 0-6) with statements concerning how well they had been understood by the system, how responsive the system was, if the system behaved as expected, and how human-like the conversational partner was. After using both versions of the system, they were also asked whether they preferred one of the versions and to what extent (1 or 2 points, which gives a maximum score of 16 to any version, when totalling all subjects).

There was no significant difference between the two versions with regard to how many of the tasks were completed successfully. However, the incremental version was clearly preferred in the overall judgement (9 points versus 1). Only one of the more specific questions yielded any significant difference between the versions: the incremental version was judged to be more human-like for the successful dialogues (5,2 on average vs. 4,5; Wilcoxon signed rank test; $p<0.05$).

The results from the evaluation are in line with what could be expected. A non-incremental system can be very efficient if the system understands the number sequence the first time, and the ASR vocabulary is in this case very limited, which explains why the success-rate was the same for both systems. However, the incremental version was experienced as more pleasant and human-like. One explanation for the better rating of the incremental version is that the acknowledgements encouraged the subjects to package the digits into installments, which helped the system to better read back the sequence using the same installments.

6 Conclusions and future work

To sum up, we have presented a dialogue system that through the use of novel techniques (incremental prosodic analysis, reactive connection between ASR and TTS, fully incremental architecture) achieves an unprecedented level of reactiveness (from a minimum latency of 750ms, as typically used in dialogue systems, down to one of 200ms), and is consequently evaluated as more natural than more typical setups by human users. While the domain we've used is relatively simple, there are no principled reasons why the techniques introduced here should not scale up.

In future user studies, we will explore which factors contribute to the improved experience of using an incremental system. Such factors may include improved responsiveness, better installment packaging, and more elaborate feedback. It would also be interesting to find out when rapid responses are more important (e.g. acknowledgements), and when they may be less important (e.g., answers to task-related questions).

We are currently investigating the transfer of the prosodic analysis to utterances in a larger domain, where similarly instructions by the user can be given in installments. But even within the currently used micro-domain, there are interesting issues still to be explored. In future versions of the system, we will let the modules pass parallel hypotheses and also improve the incremental generation and synthesis. Since the vocabulary is very limited, it would also be possible to use a limited domain synthesis (Black & Lenzo, 2000), and explore how the nuances of different backchannels might affect the dialogue. Another challenge that can be researched within this micro-domain is how to use the prosodic analysis for other tasks, such as distinguishing correction from dictation (for example if U.14 in Table 1 would not begin with a “no”). In general, we think that this paper shows that narrowing down the domain while shifting the focus to the modelling of more low-level, conversational dialogue phenomena is a fruitful path.

Acknowledgements

This work was funded by a DFG grant in the Emmy Noether programme. We would also like to thank Timo Baumann and Michaela Atterer for their contributions to the project, as well as Anna Iwanow and Angelika Adam for collecting and transcribing the data used in this paper.

References

Aist, G., Allen, J. F., Campana, E., Galescu, L., Gómez Gallo, C. A., Stoness, S. C., Swift, M., & Tanenhaus, M. (2006). Software Architectures for Incremental Understanding of Human Speech. In *Proceedings of Interspeech* (pp. 1922-1925). Pittsburgh PA, USA.

Allen, J. F., Ferguson, G., & Stent, A. (2001). An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces* (pp. 1-8).

Black, A., & Lenzo, K. (2000). Limited domain synthesis. In *Proceedings of ICSLP* (pp. 410-415). Beijing, China.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence, 47*, 139-159.

Clark, H. H. (1996). *Using language*. Cambridge, UK: Cambridge University Press.

Dohsaka, K., & Shimazu, A. (1997). System architecture for spoken utterance production in collaborative dialogue. In *Working Notes of IJCAI 1997 Workshop on Collaboration, Cooperation and Conflict in Dialogue Systems*.

Dutoit, T., Pagel, V., Pierret, N., Bataille, F., & Vreken, O. v. d. (1996). The MBROLA project: Towards a set of high-quality speech synthesizers free of use for non-commercial purposes. In *Proceedings of ICSLIP '96* (pp. 1393-1396).

Edlund, J., Gustafson, J., Heldner, M., & Hjalmarsson, A. (2008). Towards human-like spoken dialogue systems. *Speech Communication, 50*(8-9), 630-645.

Ferrer, L., Shriberg, E., & Stolcke, A. (2002). Is the speaker done yet? Faster and more accurate end-of utterance detection using prosody. In *Proceedings of ICSLP* (pp. 2061-2064).

Kilger, A., & Finkler, W. (1995). *Incremental Generation for Real-Time Applications*. Technical Report RR-95-11, German Research Center for Artificial Intelligence.

Koiso, H., Horiuchi, Y., Tutiya, S., Ichikawa, A., & Den, Y. (1998). An analysis of turn-taking and backchannels based on prosodic and syntactic features in Japanese Map Task dialogs. *Language and Speech, 41*, 295-321.

Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., Warmuth, M., & Wolf, P. (2003). The CMU SPHINX-4 speech recognition system.. In *Proceedings of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*. Hong Kong.

Levelt, W. J. M. (1989). *Speaking: From Intention to Articulation*. Cambridge, Mass., USA: MIT Press.

Liu, J., Zheng, T. F., Deng, J., & Wu, W. (2005). Real-time pitch tracking based on combined SMDSF. In *Proceedings of Interspeech* (pp. 301-304). Lisbon, Portugal.

Möller, S., Smeele, P., Boland, H., & Krebber, J. (2007). Evaluating spoken dialogue systems according to de-facto standards: A case study. *Computer Speech & Language, 21*(1), 26-53.

Raux, A., & Eskenazi, M. (2007). A multi-Layer architecture for semi-synchronous event-driven dialogue Management. In *ASRU 2007*. Kyoto, Japan..

Raux, A., & Eskenazi, M. (2008). Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In *Proceedings of SIGdial 2008*. Columbus, OH, USA.

Sacks, H., Schwegloff, E., & Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language, 50*, 696-735.

Schlangen, D., & Fernández, R. (2007). Speaking through a noisy channel: experiments on inducing clarification behaviour in human-human dialogue. In *Proceedings of Interspeech 2007*. Antwerp, Belgium.

Schlangen, D., & Skantze, G. (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-09)*. Athens, Greece.

Skantze, G., & Edlund, J. (2004). Robust interpretation in the Higgins spoken dialogue system. In *Proceedings of ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*. Norwich, UK.

Skantze, G. (2007). *Error Handling in Spoken Dialogue Systems - Managing Uncertainty, Grounding and Miscommunication*. Doctoral dissertation, KTH, Department of Speech, Music and Hearing.

Skantze, G. (2008). Galatea: A discourse modeller supporting concept-level error handling in spoken dialogue systems. In Dybkjær, L., & Minker, W. (Eds.), *Recent Trends in Discourse and Dialogue*. Springer.

Stoness, S. C., Tetreault, J., & Allen, J. (2004). Incremental parsing with reference interaction. In *Proceedings of the ACL Workshop on Incremental Parsing* (pp. 18-25).

Tanenhaus, M. K., & Brown-Schmidt, S. (2008). Language processing in the natural world. In Moore, B. C. M., Tyler, L. K., & Marslen-Wilson, W. D. (Eds.), *The perception of speech: from sound to meaning* (pp. 1105-1122).

Wirén, M. (1992). *Studies in Incremental Natural Language Analysis*. Doctoral dissertation, Linköping University, Linköping, Sweden.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann.