An Incremental System that Graphically Signals Speech Understanding

Anonymous ACL submission

Abstract

1 Introduction

Suppose a native speaker of English is listening to and transcribing the speech of a native Spanish speaker with one important consideration: the native English speaker doesn't understand Spanish. Upon inspection, it could very well be the case that the transcription is somewhat accurate; the words are segmented properly and the spelling seems to mostly be correct. This illustrates the disconnect between speech reccognition (ASR) and natural language understanding (NLU); i.e., *intent recognition*: simply being able to recognise words is only the beginning of understanding speech.

Signaling understanding is an important aspect of spoken dialogue between humans. Humas signal understanding by providing feedback such as nodding, spoken backchannels (e.g., *uh-huh*) or by performing some kind of action (Clark, 1996). Crucially, this feedback is provided as the utterance unfolds, such that installments of speech given by the speaker are confirmed as being understood before the speaker commits to continuing with speaking. Current speech-based personal assistants (PAs) don't work like this, however; rather, they require spoken input from users (i.e., speakers) as complete intents allowing the PA to signal understanding only by displaying the ASR (which, as illustrated above, does not denote *understandig*) and carrying out the requested task–the user can only determine if her intent was recognised if the correct task was carried out. Moreover, just how is a user to know what kind of things a PA could potentially understand; i.e., what are the *affordances*?[1]

[1]For example, current PAs run a query in a search engine if a request is not understood.

Current virtual personal assistants require their users to either formulate complex intents in one utterance (as in "make a phone call to Peter Miller on his mobile phone") or go through tedious subdialogues ("phone call – who would you like to call? – Peter Miller – I have a mobile number and a work number. Which one do you want?"). This is not how one would interact with a human assistant, where the request would be naturally structured into smaller chunks that individually get acknowledged ("can you make a connection for me? - sure - with Peter Miller - uh huh - on his mobile - got it"). We present a mixed voice/graphical interface that also follows this principle of "early closure on minimal information units".

To address this, some PAs attempt to learn a *user model* so as to predict what it is the user wants from the system requiring little or no spoken input from the user. These can range on a continuum of *non-predictive* systems that don't attempt to learn anything about the user, to *fully-predictive* systems (such as Google Now) that attempt to predict what a user will utter before the user even utters it (e.g., a user receives a notification on her device with the text: "Do you want directions to a French restaurant in the downtown area?").

In this paper, we present a PA that can provide feedback of understanding to the user *incrmentally* as the user's utterance unfolds. Indeed, our system allows users to make requests in installments instead of fully thought-out requests. Our system does this by displaying ongoing understanding at each recognised word to the user in an intuitive tree-like graphical interface that can be displayed on a mobile devide. We evaluate our system by directing participants to perform tasks using it under non-incremental (i.e., traditional ASR endpointing) and incremental conditions and then asking the participants to compare the two conditions with questionaires. We further compared

a non-predictive version with a version that attempted to automatically predict the intent of the participant based on previous experiences with the user. We report that the participants found the interface intuitive and easy to understand, that they recognised that at times the system responded as they were speaking, and that at times the system could predict what task they waneted the system to do with little input from the user.

The remainder of this paper is organised as follows: in the following section, we provide a discussion of related work, then describe our system in Section 3. We then explain the experiments and give the results in Section 4. We then give a final discussion, ideas for future work, and conclude.

2 Related Work

This work brings together and builds upon several threads of other previous research. (Chai et al., 2014) attempted to address misalignments in common ground (Clark and Schaefer, 1989) between systems (in their case, robots) and humans by informing the human of the internal state of the system. We take this idea and apply it a PA by displaying the internal state of the system to the user in an intuitive, tree-like structure (explained in Section 3.5), allowing the user to determine if understanding has taken place by the system. Such information presentation is a way of providing feedback and backchannels to the user. (Dethlefs et al., 2015) provides a good review of work that shows that backchannels facilitate grounding, feedback, and clarifications in human spoken dialogue. The apply an *information density* approach to determining when to backchannel using speech. Because we don't backchannel using speech here, there is no potential overlap between the human user and the system; rather, our system can display backchannels and ask clarifications without interrupting (that is, frustrating) the user.

Though different in many ways, our work is similar in some regards to (Larsson et al., 2011), which displays information to the user and allows the user to navigate the display itself (e.g., by saying *up* or *down* in a menu list), functionality that we intent to apply to our display in future work.

Some of the work here is inspired by the *Microsoft Language Understanding Intelligent Service* (LUIS) project (Williams et al., 2015). While our system by no means achieves the scale that LUIS does, we offer here an open source LUIS-like system (with the important addition of the display interface) that is authorable (using JSON files; we leave authoring using a web interface–like LUIS–to future work), extendible (affordances can be easily added), incremental (going beyond LUIS), trainable (i.e,. can learn from examples, but can still function well without examples), and can learn through interacting (left for future work, but here we apply a simplified user model that learns during interaction).

3 System Description

This section introduces and describes our SDS, which is modularised into four main components: automatic speech recognition (ASR), natural langauge understanding (NLU), dialogue management (DM), and the user interface (UI) which, as explained below, is visualised as a right-branching tree. For the remainder of this section, each module is exlpained in turn. First, however, we explain what is meant by incremental processing and the role that plays in the work presented here.

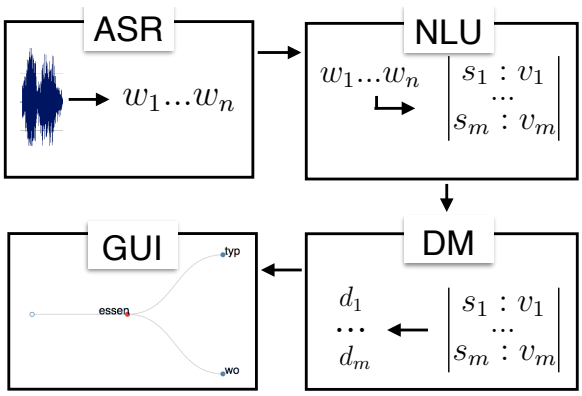

Figure 1: Overview of system made up of ASR which takes in a speech signal and produces transcribed words, NLU, which takes words and produces a slots in a frame, DM which takes slots and produces a decision for each, and the UI which displays the state of the system.

3.1 Incremental Dialogue

Of prime importance in our SDS–an aspect of our SDS that sets it apart from others–is the requirement that it processes *incrementally*. An oftencited concern with incremental processing is regarding informativeness: why act so soon when waiting (even just for a moment) would allow additional information, resutling in more-informed decisions? The trade-off here is all-important: *naturalness* as perceived by the end user who is interacting with the SDS. Indeed, it has been shown

that humans perceive incremental systems as being more natural than traditional, turn-based systems (Aist et al., 2006; Skantze and Schlangen, 2009; Skantze and Hjalmarsson, 1991; Asri et al., 2014), offer a more human-like experience for the human users (Edlund et al., 2008) and are more satisfying to interact with than non-incremental systems (Aist et al., 2007). Psycholinguistic research has also shown that humans process (i.e., comprehend) utterances as they unfold and do not wait until the end of an utterance to begin the comprehension process (Tanenhaus, 1995; Spivey et al., 2002).

The trade-off between informativeness and naturalness can be reconciled when mechanisms are in place where earlier desicions can be repaired. Such mechanisms were introduced in the incremental unit (IU) framework for SDS (Schlangen and Skantze, 2009; Schlangen and Skantze, 2011). Following (Kennington et al., 2014a), SDSs based on the IU-network approach consist of a network of processing *modules*. A typical module takes input from its *left buffer*, performs some kind of processing on that data, and places the processed result onto its *right buffer*. The data are packaged as the payload of *incremental units* (IUs) which are passed between modules. The IUs themselves are also interconnected via *same level links* (SLL) and *grounded-in links* (GRIN), the former allowing the linking of IUs as a growing sequence, the latter allowing that sequence to convey what IUs directly affect it. A complication particular to incremental processing is that modules can "change their mind" about what the best hypothesis is, in light of later information, thus IUs can be *added*, *revoked*, or *committed* to a network of IUs.

The modules exlpained in the remainer of this section are implemented as IU-modules and process incrementally. Each will now be explained.

3.2 Speech Recognition

Incremental processing begins with modules that take in input; in the case of our SDS, that is the ASR component. Incremental ASR must transcribe uttered speech into words and words must be forthcoming from the ASR as early as possible (i.e., the ASR must not wait for endponiting in order to act). Each module that follows must also process incrementally, acting in lock-step upon input as it is received. Incremental ASR is not new (Baumann et al., 2009) and many of the current freely-accessible ASR systems can produce output (semi-) incrementally.

In our SDS, we opt for Google ASR because of its wide vocbaulary coverage of the language we are interested in (German). We are able to package ASR output from the Google service into IUs as explained above. Those word IUs are passed to the NLU module, which will now be explained.

3.3 Language Understanding

We approach the task of NLU as a slot-filling task (a very common approach; see (Tur et al., 2012)) where the system can fill the task when all slots of a frame are filled. The main driver of the NLU in our SDS is the SIUM model of NLU introduced in (Kennington et al., 2013). SIUM has been used in several systems which have reported impressive results in various domains, languages, and tasks (Kennington et al., 2014b; Kennington et al., 2015). Though originally a model of reference resolution, the authors hinted that it could be used for general NLU, which we do here. The model is formalised as follows:

$$P(I|U) = \frac{1}{P(U)} P(I) \sum_{r \in R} P(U|R) P(R|I) \quad (1)$$

That is, $P(I|U)$ is the probability of the intent I (i.e., a frame slot) behind the speaker's (ongoing) utterance U. This is recovered using the mediating variable R, a set of *properties* which map between aspects of U and aspects of I. These properties could be visual properties of visible objects, or they could be more abstract properties that intents might have, which we opt for here (e.g., the intent of a `restaurant` might be filled by a certiain type of cusine such as `italian` which has (among others) properties like `pasta`, `mediteranian`, `vegetarian`, etc.). Properties are pre-defined by a system designer and can match words that might be uttered to describe the intent in question. The mapping betwen properties and aspects of U can be learned from data. During application, $P(U|R)$ can produce a distribution over words (or properties, see below) which are summed over and the probability mass for each property is accumulated for each intent, resulting in a distribution over possible intents. This occurs at each word increment, where the distribution from the previous increment is combined via $P(I)$, keeping track of the distribution over time.

In our SDS, we apply an instantion of SIUM for each slot (explained in Section 4), all of which update at each word increment. At each word increment, the updated slots (and their corresponding) distributions are given to the DM, which will now be explained.

3.4 Dialogue Manager

The primary job of our DM is to determine *when* to act, given the unfolding utterance. That is, at each word, the DM needs to choose one of the following:

- `wait` – don't act until more information is forthcoming
- `select` – the NLU is confident enough that a slot can be filled
- `request` – the dialogue has reached a state where the system has asked for a (yes/no) clarification request
- `confirm` – the user has responded to the clarification request (which acts similar to a `select`)

This is a crucial part to our SDS which sets it apart from other systems in that the DM is called upon at each word to decide *when* to act, rather than *how* to act, effecively giving the DM the control over timing of actions rather than relying on ASR endpointing. The DM policy is based on a confidence score (CONF) derived from the NLU (in this case, we used the distribution's argmax value) suing thresholds for `wait`, `confirm` and `select` set by hand (i.e., trial and error). Though the thresholds were statically set, we applied OpenDial (Lison, 2015) as a modularised iu-module to perform the task of the DM with the future goal that these values could be adjusted through reinforcement learning (which OpenDial could provide). The DM processes and makes a decision for *each slot*, with the assumption that only one slot out of all that are processed will result in an non-`wait` action (though this is not constrained). The candidate slots that are processed depends on the state of the UI (described below); only slots represented by visible nodes are considered, thereby reducing the possible frames that could be predicted.

3.5 Graphical Interface

The goal of the display is to inform the user about the internal state of the ongoing understanding. One motivation for this is that the user can determine if the system understood the user's intent before providing the user with a response (e.g., a list of restaurants of a certain type)–i.e., if any misunderstanding takes place, it happens before the system commits to an action and is potentially more easily repaired.

The display is a right-branching tree, where the branches directly off the root node display the affordances of the system (i.e., what domains of things it can understand and do something about). When the first tree is displayed, it represents a state of the NLU where none of the slots are filled, an example of which is shown in Figure 2.

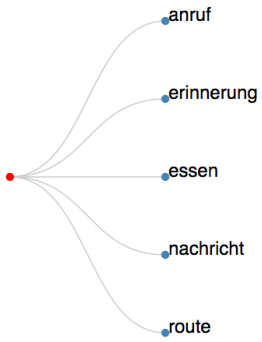

Figure 2: Example tree as branching from the root; each branch represents a system affordance (i.e., making a phone call, reminder, finding a restaurant, leaving a message, and finding a route).

When a user selects a domain to ask about, the tree is adjusted to make that domain the only one displayed and the slots that are required for that domain are shown as branches. The user can then fill those slots (i.e., branches) by uttering the name of the slot, or, alternatively, by uttering the item to fill slot directly. For example, at a minimum, the user could utter the name of the domain then an item for each slot (e.g., *food French downtown*) or the speech could be more natural (e.g., *I'm quite hungry, I am looking for some French food maybe in the downtown area*). When something is uttered that falls into th `confirm` state of the DM as explained above, the display expands the subtree under question (*//todo: bring QUD into this?//*) and marks the item with a question mark. And example of this is shown in Figure 3 where *französisch* (French) is requested to be confirmed. At this point, the user can utter any kind of confirmation. A positive confirmation would fill the slot with the item in question, collapsing that particular branch of the tree. A negative confirmation would re-

tract the question, but leave the branch expanded. The expanded branches are displayed according to their rank as given by the NLU's probability distribution.

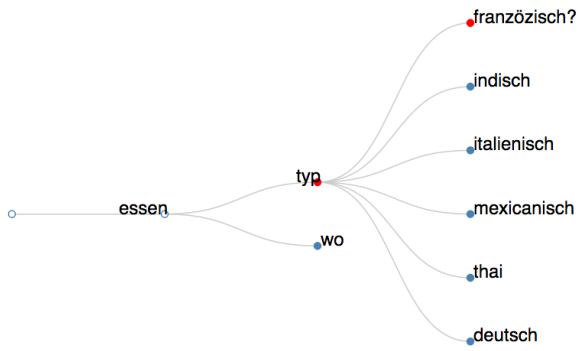

Figure 3: Example tree asking for confirmation on a specific node (in red with a question mark).

A filled branch is collapsed, visually marking it as filled. At any time, a user can backtrack by saying *no* (or equivalent) or start the entire interaction over from the beginning with a keyword, e.g., *restart*. To aid the user's attention, the node under question is marked in red, where filled slots are represnted by blue nodes, and filled nodes represent candidates for the current slot in question. For cases where the system is in the `wait` state for several words (during which there is no change in the tree), the system signals activity at each word by causing the red node in question to temporarily change to white, then back to red (i.e., appearing as a blinking node to the user). Figure 4 represents a filled frame represnted as tree with one branch for each filled slot.

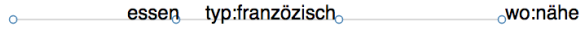

Figure 4: Example tree where all of the slots are filled. (i.e., `domain:food`, `location:nearby`, `type:french`)

Such an interface clearly shows the internal state of the SDS; whether or not it has understood the request so far. It is desinged to aid the user's attention to the slot in question, and clearly indicates the affordances that the system has. At the moment the interface is simply a read-only display that is purely speech-driven, but it could be augmented with additional funcionalities, such as tapping a node for expansion or typing intput that the system might not yet display. It is currently implemented as a web-based interface (using the javascript D3 library), allowing it to be usable as a web application on any machine or mobile device.

4 Experiments

In this section, we describe two experiments in which we evaluated our system. In order to best evaluate our system, we recruited participants to interact with our system in varied settings. We will describe how the data were collected from the participants, then explain each experiment and give results.

4.1 Task & Instructions

The participants were seated at a desk and given written instructions that they were to use the system to perform as many tasks as possible in the allotted time. The instructions gave them several examlpes of kinds of tasks the system could understand with an icon representing the domain and then text for the rest of the task, as shown in Figure 5. In front of the participant towards the rear of the table was a computer screen that would show the task in the middle of the screen. In front of that screen closer to the participant was a small mobile tablet that showed the UI.[2] The user was instructed to convey the task presented on the screen to the system such that the UI on the tablet would have a completed tree (i.e., the tree was in its filled state as in Figure 4). When the participant was satisfied that the system understood her intent, she was to press spacebar which triggered a new task to be displayed and reset the tree to its start state (which was the root node linking to the 5 possible tasks).

The possible tasks were *call*, which had a single slot for *name* to be filled (i.e., one out of the 22 most common German given names); *message* which had a slot for *name* and a slot for the *message* (which, when invoked, would simply fill in directly from the ASR until 1 second of silence

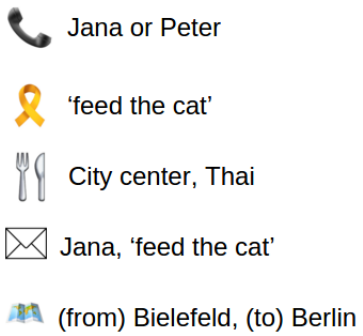

Figure 5: Examples of tasks, as presented to each participant. Each icon represents a specific task domain (e.g., *route* finding).

[2]We used a Samsung 8.4 Pro turned to its side to show a bigger width for the tree to grow to the right.

was detected); *eat* which had slots for *type* (in this case, 6 possible types) and *location* (in this case, 6 possible known locations based around the city of Bielefeld); *route* which had slots for *source* city and the *destination* city; and *reminder* which had a slot for *message*. Some of the slot types were shared across domains, such as *message* (across *reminder* and *message*), *name* (across *message* and *call*), as well as *source* and *destination* which shared the same list of German city names (the top 100 most populous cities).

The tasks presented to the participant were randomly chosen: first, the domain was randomly chosen from the 5 possible domains, and then each slot value to be filled was randomly chosen (the *message* slot for the *name* and *messag* domains was randomly selected from a list of 6 possible "messages", each with 2-3 words; e.g., *feed the cat*, *visit grandma*, etc.). The system kept track of which tasks were already presented to the participant. At any time after the first task, the system could choose a task that was previously presented and presented it again to the participant (with a 50% chance) so the user would often see tasks that she had seen before (with the assumption that humans who use PAs often do perform similar, if not the same, tasks more than once).

The participant was told that she would interact with the system in three different phases, each for 4 minutes, and was instructed to accomplish as many tasks as possible in that time allotment. The participant was not told what the diffrent phases were, only that the system was somewhat different in each phase. The experiments described in Sections 4.2 and 4.4 respectively describe and report a comparison first between the first and second phase (denoted the non-incremental and incremental variants of the system) and a comparison between the incremental and incremental-adaptive phases. Each of these phases are described below. Before the participant began Phase 1, they were able to try it out for up to 4 minutes (in Phase 1 settings) and ask questions about how it worked, allowing them to get used to the interface before the actual experiment began.

Phase 1: Non-incremental In this phase, the system did not appear to work incrementally; that is, the system only displayed tree updates after ASR endpointing (of 1.2 seconds, which is a resonable amount of time to expect a response from a commercial spoken PA). The system displayed the ongoing ASR on the tablet as it was recognised (as is often done in commercial PAs). During speaking, the tree would not be displayed on the tablet. The participant knew that the system fully understood the task when the displayed tree had no more unfilled branches (as in Figure 4).

Phase 2: Incremental In this phase, the system displayed the tree information incrementally as explained above. The ASR was no longer displayed; only the tree provided feedback in understanding.

After Phase 2, a 10-question questionnaire was displayed on the screen for the participant to fill out comparing Phase 1 and Phase 2. For each question, they had the choice of *Phase 1*, *Phase 2*, *Both*, and *Neither*. (See appendix for full list of questions.)

Phase 3: Incremental-adaptive In this phase, the incremental system was again presented to the participant with an added user model that "learned" about the user. If the user saw a task more than twice, the user model would predict that, if the user chose that task domain again (e.g., *route*) then the system would automatically ask a clarification on the slots. If the user saw a task more than 3 times, the system skipped asking for clarifications and filled in the domain slots comlpetely, requiring the user only to press the spacebar to confirm it was the correct one (i.e., to end the task). An example progression might be as follows: a participant is presented with the task *route from Bielefeld to Berlin*, then the user would attempt to get the system to fill in the tree (i.e., slots) with those values. After some interaction in other domains, the user sees the same task again. After additional interaction in other domains, the task is repeated and here the user must only say "yes" for each slot to confirm the system's prediction. Later, if the task is presented a fourth time, the user would enter that domain (i.e, *route*) and the two slots would already be filled. If later a new route task was presented (e.g., *route from Bielefeld to Hamburg* the system would already have the slots filled for `from:Bielefeld`, `destination:Berlin`, but the user could backtrack by saying "no, to Hamburg" which would trigger the system to fill `destination:Hamburg`. Later interactions within the *route* domain would ask for a clarification on the *destination* slot since it has had several

possible values given by the participant.

After Phase 3, the participants were presented with another questionnaire on the screen to fill out with the same questions (plus two additional questions), this time comparing Phase 2 and Phase 3. For each question, they had the choice of *Phase 2*, *Phase 3*, *Both*, and *Neither*.

At the end of the three phases and questoinnaries, the participants were given a final questionnaire to fill out by hand on their general impressions of the systems.

4.2 Experiment 1: Non-Incremental vs. Incremental

4.3 Results

4.4 Experiment 2: Incremental vs. Incremental-Adaptive

4.5 Results

Appendix

The following questions were asked on both questoinnaries following Phase 2 and Phase 3 (comparing the two most latest used system versions):

- The interface useful and easy to understand.
- The assistant was easy and intuitive to use.
- The assistant understaood what I wanted to say.
- I always understood what the system wanted from me.
- The assistant made many mistakes.
- The assistant did not respond while I spoke.
- It was sometimes unclear to me if the assistant understood me.
- The assistant responde while I spoke.
- The assistant sometimes did things that I did not expect.
- When the assistant made mistakes, it was easy for me to correct them.

In addition to the above 10 questions, the following were also asked on the questionnaire following Phase 3:

- I had the feeling that the assistant attempted to learn about me.
- I had the feeling that the asssitant made incorrect guesses.

The following questions were used on the general questoinnaire (as translated into English):

- I regularly use peronal assistants such as Siri, Corana, Google now or Amazon Echo: Yes/No
- I have never used a speech-based personal assistant: Yes/No
- What was your general impression of our personal assistants?
- Would you use one of these assistants on a smartphone or tablet if it were available? If yes, which one?
- Do you have suggestions that you think would help us improve our assistants?
- If you have used other speech-based interfaces before, do you prefer this interface?

References

Gregory Aist, James Allen, Ellen Campana, Lucian Galescu, Carlos A. Gomez Gallo, Scott Stoness, Mary Swift, and Michael Tanenhaus. 2006. Software architectures for incremental understanding of human speech. In *Interspeech 2006*, pages 1922—-1925.

Gregory Aist, James Allen, Ellen Campana, Carlos Gomez Gallo, Scott Stoness, and Mary Swift. 2007. Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods. In *Pragmatics*, volume 1, pages 149–154, Trento, Italy.

Layla El Asri, Romain Laroche, Olivier Pietquin, and Hatim Khouzaimi. 2014. NASTIA: Negotiating Appointment Setting Interface. In *Proceedings of LREC*, pages 266–271.

Timo Baumann, Michaela Atterer, and David Schlangen. 2009. Assessing and Improving the Performance of Speech Recognition for Incremental Systems. In *Proceedings of NAACL-HLT 2009*, Boulder, USA, jun.

Joyce Y Chai, Lanbo She, Rui Fang, Spencer Ottarson, Cody Littley, Changsong Liu, and Kenneth Hanson. 2014. Collaborative effort towards common ground in situated human-robot dialogue. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 33–40, Bielefeld, Germany.

Herbert H Clark and Edward F Schaefer. 1989. Contributing to Discourse. *Cognitive Science*, 13:259–294.

Herbert H Clark. 1996. *Using Language*. Cambridge University Press.

Nina Dethlefs, Helen Hastie, Heriberto Cuayáhuitl, Yanchao Yu, Verena Rieser, and Oliver Lemon. 2015. Information density and overlap in spoken dialogue. *Computer Speech and Language Language*, 37:82–97.

Jens Edlund, Joakim Gustafson, Mattias Heldner, and Anna Hjalmarsson. 2008. Towards human-like spoken dialogue systems. *Speech Communication*, 50(8-9):630–645.

Casey Kennington, Spyros Kousidis, and David Schlangen. 2013. Interpreting Situated Dialogue Utterances: an Update Model that Uses Speech, Gaze, and Gesture Information. In *Proceedings of SIGdial*.

Casey Kennington, Spyros Kousidis, and David Schlangen. 2014a. InproTKs: A Toolkit for Incremental Situated Processing. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 84–88, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Casey Kennington, Spyros Kousidis, and David Schlangen. 2014b. Situated Incremental Natural Language Understanding using a Multimodal, Linguistically-driven Update Model. In *Proceedings of CoLing*.

Casey Kennington, Ryu Iida, Takenobu Tokunaga, and David Schlangen. 2015. Incrementally Tracking Reference in Human/Human Dialogue Using Linguistic and Extra-Linguistic Information. In *Proceedings of NAACL-HLT*, Denver, U.S.A. Association for Computational Linguistics.

Staffan Larsson, Alexander Berman, and Jessica Villing. 2011. Multimodal Menu-based Dialogue with Speech Cursor in DICO II +. In *Computational Linguistics*, number June, pages 92–96.

Pierre Lison. 2015. A hybrid approach to dialogue management based on probabilistic rules. *Computer Speech & Language*, 34(1):232–255.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics on EACL 09*, 2(1):710–718.

David Schlangen and Gabriel Skantze. 2011. A General, Abstract Model of Incremental Dialogue Processing. In *Dialogue & Discourse*, volume 2, pages 83–111.

Gabriel Skantze and Anna Hjalmarsson. 1991. Towards Incremental Speech Production in Dialogue Systems. In *Word Journal Of The International Linguistic Association*, pages 1–8, Tokyo, Japan, sep.

Gabriel Skantze and David Schlangen. 2009. Incremental dialogue processing in a micro-domain. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics on EACL 09*, (April):745–753.

Michael J Spivey, Michael K Tanenhaus, Kathleen M Eberhard, and Julie C Sedivy. 2002. Eye movements and spoken language comprehension: Effects of visual context on syntactic ambiguity resolution. *Cognitive Psychology*, 45(4):447–481.

Michael Tanenhaus. 1995. Integration of Visual and Linguistic Information in Spoken Language Comprehension. *Science*, 268:1632–1634.

Gokhan Tur, Li Deng, Dilek Hakkani-Tür, and Xiaodong He. 2012. Towards deeper understanding: Deep convex networks for semantic utterance classification. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 5045–5048. IEEE.

Jason D Williams, Eslam Kamal, Mokhtar Ashour, Hani Amr, Jessica Miller, and Geoff Zweig. 2015. Fast and easy language understanding for dialog systems with Microsoft Language Understanding Intelligent Service (LUIS). In *Proceedings of 2015 SIGDIAL Conference, Prague*. ACL – Association for Computational Linguistics, sep.