# Supporting Spoken Assistant Systems with a Graphical User Interface that Signals Incremental Understanding and Prediction State

**Casey Kennington** and **David Schlangen**
DSG / CITEC / Bielefeld University

## Abstract

Arguably, spoken dialogue systems these days are most often used not in hands-busy or eyes-busy situations but rather in settings where a graphical display is also available, namely on a mobile phone. We explore the use of a graphical output modality for signalling incremental understanding and prediction state. By visualising the current dialogue state and possible continuations of it as a simple tree, and allowing interaction with that visualisation (e.g., for confirmations or corrections), the system provides both feedback on past user actions and guidance on possible future ones, and it can span the continuum from full prediction of user intent (as in systems like GoogleNow) to full slot filling. We evaluate our system with real users and report that they found the system intuitive and easy to use, and that incremental and adaptive settings allow users to accomplish tasks more effecively.

## 1 Introduction

Current virtual personal assistants (PAs) require 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 sub-dialogues (e.g., "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 (e.g., "Can you make a connection for me?" – *sure* – "with Peter Miller" - *uh huh* - "on his mobile" - *dialing now*). Current PAs only signal ongoing understanding by displaying the state of the recognised speech (ASR) to the user, but not their semantic interpretation of it.

Another type of assistant system forgoes enquiring user intent altogether and infers *likely* intents from context. GoogleNow for example might present traffic information to a user picking up their mobile phone at their typical commute time. These system display their "understanding" state, but do not allow any type of interaction with it, apart from dismissing the provided information.

In this work, we explore adding a graphical interface modality that makes it possible to see these interaction styles as extremes on a continuum, and to realise positions between these extremes. We present a mixed graphical/voice enabled PA that can provide feedback of understanding to the user *incermentally* as the user's utterance unfolds–allowing users to make requests in installments instead of fully thought-out requests. It does this by displaying ongoing understanding at each recognised word to the user in an intuitive tree-like GUI that can be displayed on a mobile device. We evaluate our system by directing users to perform tasks using it under non-incremental (i.e., ASR endpointing) and incremental conditions and then asking the users to compare the two conditions with questionaires. We further compared a non-adaptive with an adaptive version of our system. We report that the users found the interface intuitive and easy to understand and that users were able to perform tasks more efficiently with incremental as well as adaptive variants of the system.

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 understanding (i.e., 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 (through speech from the robot). We take this idea and apply it to a PA by displaying the internal state of the system to the user via a GUI (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. (2016) provides a good review of work that shows that backchannels facilitate grounding, feedback, and clarifications in human spoken dialogue, and 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 frustrating the user through inadvertent overlaps.

Though different in many ways, our work is similar in some regards to that of 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 intend 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 as an additional contribution an open source LUIS-like system (with the important addition of the graphical interface) that is authorable (using JSON files; we leave authoring using a web interface like that of LUIS to future work), extensible (affordances can be easily added), incremental (in that respect going beyond LUIS), trainable (i.e,. can learn from examples, but can still function well without examples), and can learn through interacting (here we apply a user model that learns during interaction).

## 3 System Description

This section introduces and describes our SDS, which is modularised into four main components: ASR, natural language understanding (NLU), dialogue management (DM), and the user interface (GUI) which, as explained below, is visualised as a right-branching tree. The overall system is represented in Figure 1. For the remainder of this section, each module is explained in turn. As each module processes input incrementally (i.e., word for word), we first explain our framework for incremental processing.

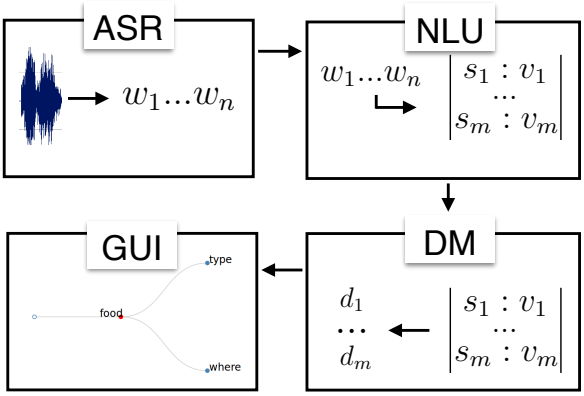


**Figure 1:** Overview of system made up of ASR which takes in a speech signal and produces transcribed words, NLU, which takes words and produces a slots in a frame, DM which takes slots and produces a decision for each, and the GUI which displays the state of the system.

### 3.1 Incremental Dialogue

An aspect of our SDS that sets it apart from others is the requirement that it process *incrementally*. One potential concern with incremental processing is regarding informativeness: why act early when waiting might provide additional information, resulting in better-informed decisions? The trade off is *naturalness* as perceived by the user who is interacting with the SDS. Indeed, it has been shown that human users 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 (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 and Spivey-Knowlton, 1995; Spivey et al., 2002).

The trade-off between informativeness and naturalness can be reconciled when mechanisms are

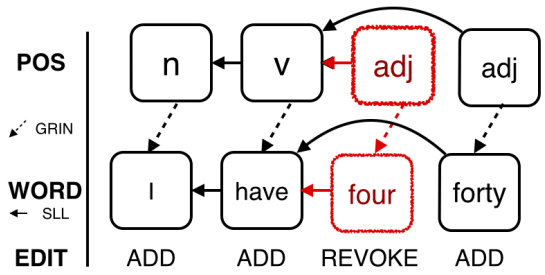


**Figure 2:** Example of IU network; part-of-speech tags are grounded into words, tags and words have same level links with left IU; *four* is revoked and replaced with *forty*.

in place that allow earlier desicions can be repaired. Such mechanisms are offered by the incremental unit (IU) framework for SDS (Schlangen and Skantze, 2011), which we apply here. Following Kennington et al. (2014a), incremental systems consist of a network of processing *modules*. A typical module takes input, performs some kind of processing on that data, and produces output. The data are packaged as the payload of *incremental units* (IUs) which are passed between modules. The IUs themselves are interconnected via so-called *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 (see Figure 2 for an example of incremental ASR). Thus IUs can be *added*, but can be later *revoked* and replaced in light of new information. This allows incremental components to take advantage of up-to-date information, but have the potential to function in such a way that users perceive the system as more natural.

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

### 3.2 Speech Recognition

The module that takes speech input from the user in our SDS is the ASR component. Incremental ASR must transcribe uttered speech into words which must be forthcoming from the ASR as early as possible (i.e., the ASR must not wait for endponiting to produce output). 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 vocabulary coverage of the language we are evaluating 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 an intent is complete 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 substantial results in various domains, languages, and tasks (Kennington et al., 2014b; Kennington et al., 2015; Kennington and Schlangen, 2016) Though originally a model of reference resolution, it was always intended to 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 $I$. We opt for abstract properties 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 which are summed over and the probability mass for each property is accumulated for each intent, resulting in a distribution over possible intents.[1] 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.

To allow our system to function with minimal amounts of training data, we apply a simple rule: if some $r \in R$ and $w \in U$ are such that $r = w$ (e.g., an intent has the property of `pasta` and the word *pasta* is uttered, then $P(U = w|R = r) = 1$.

[1] In Kennington et al. (2013) the authors apply Bayes' Rule to allow $P(U|R)$ to produce a distribution over properties, which we adopt here.

In our SDS, we apply an instantion of SIUM for each slot, 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 with the argmax for this slot from NLU.
- `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) using 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 an 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 GUI (described below); only slots represented by visible nodes are considered, thereby reducing the possible frames that could be predicted.

### 3.5 Graphical User Interface

The goal of the GUI 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 afforances 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 3.

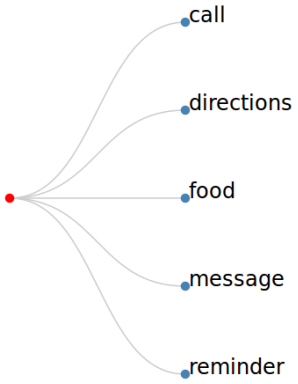


**Figure 3:** Example tree as branching from the root; each branch represents a system affordance (i.e., making a phone call, reminder, finding a restaurant, leaving a message, and finding a route).

When a user verbally 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 displayed name, or, alternatively, by uttering the item to fill the 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 Thai downtown*) or the speech could be more natural (e.g., *I'm quite hungry, I am looking for some Thai food maybe in the downtown area*). When something is uttered that falls into the `request` 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 (see Figure 4). At this point, the user can utter any kind of confirmation. A positive confirmation fills the slot with the item in question. A negative confirmation retracts the question, but leaves the branch expanded. The expanded branches are displayed according to their rank as given by the NLU's probability distribution. Though a branch in the display can theoretically display an unlimited number of children, we opted to only show 7

children; if a branch had more than 7 children, the bottom-most child is displayed as an ellipsis.

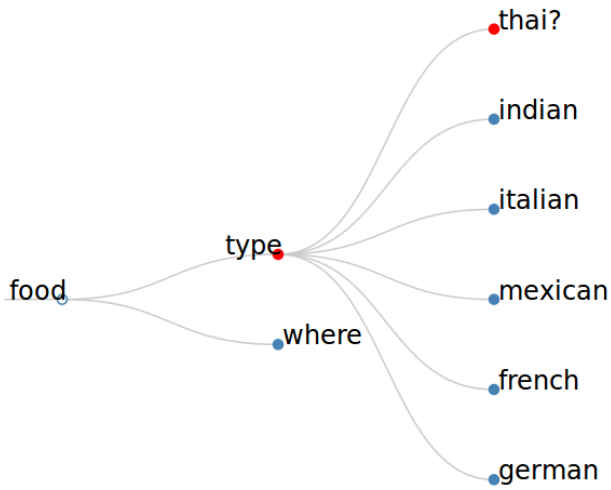


**Figure 4:** Example tree asking for confirmation on a specific node (in red with a question mark).

A completed branch is collapsed, visually marking its corresponding slot 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 5 shows a completeld filled frame, represented as tree with one branch for each filled slot.

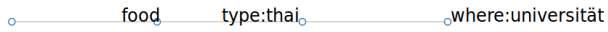


**Figure 5:** Example tree where all of the slots are filled. (i.e., `domain:food`, `location:nearby`, `type:french`)

Such an interface clearly shows the internal state of the SDS and 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. The interface is currently 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 where we evaluated our system. It is our primary goal to show that our GUI is useful and signals understanding to the user. We also wish to show that incremental presentation of such a GUI is more effective than an endpointed system. We further want to show that an adaptive system is more effective than a non-adpative system (though both would process incrementally). In order to best evaluate our system, we recruited participants to interact with our system in varied settings to compare endpointed (i.e., non-incremental) and non-adaptive as well as adaptive (though still incremental) versions. We will describe how the data were collected from the participants, then explain each experiment and give results.

### 4.1 Task & Procedure

The participants were seated at a desk and given written instructions indicating that they were to use the system to perform as many tasks as possible in the allotted time. Figure 6 shows some example tasks as they would be displayed (one at a time) to the user. A screen, tablet, and keyboard were on the desk in front of the user (see Figure 7).[2] The user was instructed to convey the task presented on the screen to the system such that the GUI on the tablet would have a completed tree (i.e., as in Figure 5). When the participant was satisfied that the system understood her intent, she was to press spacebar on the keyboard which triggered a new task to be displayed and the screen and reset the tree to its start state on the tablet (which was the

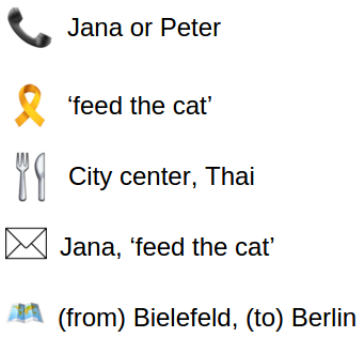


**Figure 6:** Examples of tasks, as presented to each participant. Each icon represents a specific task domain (i.e., call, reminder, find a restaurant, leave a message, or directions).

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

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

For each task, the domain was first 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 *message* 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 present 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).

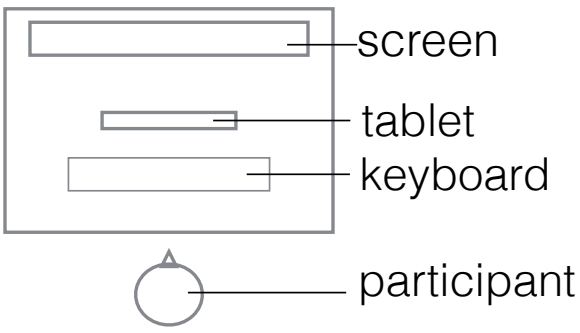


**Figure 7:** Bird's eye view of the experiment: the participant sat at a table with a screen, tablet, and keboard in front of them.

The participant was told that she would interact with the system in three different phases, each for 4 minutes, and to accomplish as many tasks as possible in that time allotment. The participant was not told what the diffrent phases were. The experiments described in Sections 4.2 and 4.3 respectively describe and report a comparison first between the Phase 1 and 2 (denoted as the *end-pointed* and *incremental* variants of the system) and a comparison between Phase 2 and 4 (*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. After this trial phase, the experiment began. For each phase, the participant knew that the system fully understood the task when the tree had no more unfilled branches (as in Figure 5).

**Phase 1: Non-incremental** In this phase, the system did not appear to work incrementally; i.e., 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).

**Phase 2: Incremental** In this phase, the system displayed the tree information incrementally without endpointing. 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 once, 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 using the previously used values. If the user saw a task more than 3 times, the system skipped asking for clarifications and filled in the domain slots completely, 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 and here the

user must only say "yes" for each slot to confirm the system's prediction. Later, if the task is presented a third 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 item, 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.

We recruited 14 participants to participate in the evaluation. We used the Mint tools data collection framework described in Kousidis et al. (2012) to log the interactions. Due to some technical issues, one of the participants did not log interactions, we collected data from 13 participants, post-Phase 2 questionnaires from 12 participants, post-Phase 3 questionnaires from all 14 participants, and general questoinnaires from all 14 participants. In the experiments that follow, we report object and subjective measures to determine the settings that produced superior results.

**Metrics** We report the subjective results of the participant questionnaires. We only report those items that were statistically significant (see Appendix for a list of the questions). We further report objective measures for each system variant: total tasks, fully correct frames, average frame fscore, and average time elapsed (averages are taken over all participants for each variant; we only used the 10 participants who fully interacted with all three phases). Discussion of the two experiments is left to the end of this section.

### 4.2 Experiment 1: Endpointed vs. Incremental

In this section we report the results of the evaluation between the *endpointed* (i.e., non-incremental; Phase 1) vs the incremental (Phase 2) variants of our system.

| | **endpointed** | **incremental** | **adaptive** |
|---|---|---|---|
| **tasks** | 105 | 122 | 124 |
| **frames** | 46 | 46 | 59 |
| **fscore** | 0.81 | 0.74 | 0.80 |
| **time** | 19.1 | 19.6 | 19.5 |

**Table 1:** Objective measures for Experiments 1 & 2: count of completed tasks, number of fully correct frames, average fscore (over all participants), and average elapsed time per task (over all participants).

**Subjective Results** We applied a multinomial test of significance to the results, treating all four possible answers as equally likely (with Bonferroni correction of 10). The item *The interface useful and easy to understand* with the answer of *Both* was significant ($\chi^2$ (4, N = 12) = 9.0, $p < .005$), as was *The assistant was easy and intuitive to use* also with the answer *Both* ($\chi^2$ (4, N = 12) = 9.0, $p < .005$). The item *I always understood what the system wanted from me* was also answered *Both* significanly more times than other answers ($\chi^2$ (4, N = 14) = 9.0, $p < .005$), similarly for *It was sometimes unclear to me if the assistant understood me* with the answer of *Both* ($\chi^2$ (4, N = 12) = 10.0, $p < .005$). These responses tell us that though the participants did not report preference for either system variant, they reported a general positive impression of the GUI (in both variants). This is a nice result; the GUI could be used in either system with benefit to the users.

**Objective Results** The *endpointed* (Phase 1) and *incremental* (Phase 2) columns in Table 1 show the results of the objective evaluation. Though the average time per task and fscore for the endpointed variant are better than those of the endpointed variant, the total number of tasks for the incremental variant was higher.

### 4.3 Experiment 2: Incremental vs. Incremental-Adaptive

In this section we report results for the evaluation between the *incremental* (Phase 2) and *incremental-adaptive* (henceforth just *adaptive*; Phase 3) systems.

**Subjective Results** We applied a multinomial test of significance to the questionnaire responses, treating all four possible answers as equally likely (with Bonferroni correction of 12). The item *The*

*interface useful and easy to understand* was answered *Both* significantly more than other answers ($\chi^2$ (4, N = 14) = 10.0, p $<$ .0042), The item *I had the feeling that the assistant attempted to learn about me* was answered significantly with *Neither* ($\chi^2$ (4, N = 14) = 8.0, p $<$ .0042), though the only other marked answer was *Phase 3* (6 times), showing that either the participants didn't notice that either phase was attempting to apply a user model at all, but if they did notice, only Phase 3 was marked. All other items were not significant.

Here again we see that there is a general positive impression of the GUI under all conditions. If anyone noticed that a system variant was attempting to learn a user model at all, they noticed that it was in Phase 3, as expected.

**Objective Results** The *incremental* (Phase 2) and *adaptive* (Phase 3) columns in Table 1 show the results for the objective evaluation for this experiment. Here, there is a clear difference between the two variants, with the adaptive showing more completed tasks, more fully correct frames, a higher fscore (all three possibly due to the fact that frames were potentially pre-filled).

### 4.4 Discussion

While the questionnaires don't express any preference for a particular system variant, the overall impression of the GUI was positive. The objective measures show that there are gains to be made when the system signals understanding at a more fine-grained interval than at the utterance level, due to the higher number of completed tasks. There are further gains to be made when the system applies even simple user modeling (i.e., adaptivity) by attempting to predict what the user might want to do given a chosen domain; allowing the system to fill the frames decreases the possibility of user error, allowing the system to successfully complete more tasks in a shorter amount of time.

The open-ended questionnaire sheds additional light. Most of the suggestions for improvement related to ASR misrecognition, not the system itself, also that the ASR reponses were too slow. Two participants suggested an ability to add "free input" as to override the ASR or select alternatives from the list. Two participants suggested that the system be more responsive (i.e., more clearly show that the system was working even if the GUI did not change) and give more feedback (i.e., backchannels) more often. For those participants that expressed preference to the non-incremental system (Phase 1), none of them had used a speech-based PA before, whereas those that expressed preference to the incremental versions (Phases 2 and 3) have used speech-based PAs before, and use them regularly. We conjecture that people without experience like to know how well the ASR is working and equate understanding with ASR, whereas those that are more familiar with PAs know that perfect ASR doesn't necessarily translate to perfect understanding–hence the need for an interface like the one presented here. A simple remedy would be to display ASR with the tree, to show the user that understanding can take place despite ASR errors.

## 5 Conclusion & Future Work

Given the subjective and object results and analysis, we conclude that an intuitive presentation that signals a system's ongoing understanding benefits end users who perform simple tasks which might be performed by a PA. The GUI that we provided, using a right-branching tree, worked well; indeed, the participants who used it found it intiutive and easy to understand. There are gains to be made when the system signals understanding at finer-grained levels than just at the end of a pre-formulated utterance. There are further gains to be made when a PA attempts to learn (even a rudimentary) user model to predict what the user might want to do next.

For future work, we intend to provide simple authoring tools for the system to make building simple PAs using our GUI easy. We want to improve the NLU and scale to larger domains.[3] We also plan on implementing this as a standalone application that could be run on a mobile device, which could actually perform the tasks.

## Appendix

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

- The interface useful and easy to understand.
- The assistant was easy and intuitive to use.

[3] Our chosen NLU approach can scale fairly well, but the GUI has some limits when applied to larger domains with thousands of items. We leave improved scaling to future work.

- 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:

- 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 Gallo, Scott Stoness, Mary Swift, and Michael Tanenhaus. 2006. Software architectures for incremental understanding of human speech. In *Proceedings of CSLP*, 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 Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 380–388, 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(2):259–294.

Nina Dethlefs, Helen Hastie, Heriberto Cuay??huitl, Yanchao Yu, Verena Rieser, and Oliver Lemon. 2016. Information density and overlap in spoken dialogue. *Computer Speech and 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 and David Schlangen. 2016. A Simple Generative Model of Incremental Reference Resolution in Situated Dialogue. *Comptuer Speech & Language*.

Casey Kennington, Spyros Kousidis, and David Schlangen. 2013. Interpreting Situated Dialogue Utterances: an Update Model that Uses Speech, Gaze, and Gesture Information. In *SIGdial 2013*, number August, pages 173–182.

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 *CoLing 2014*, pages 1803–1812.

Casey Kennington, Ryu Iida, Takenobu Tokunaga, and David Schlangen. 2015. Incrementally Tracking Reference in Human / Human Dialogue Using Linguistic and Extra-Linguistic Information. In *HLT-NAACL 2015 - Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings of the Main Conference*, pages 272–282, Denver, U.S.A. Association for Computational Linguistics.

Spyridon Kousidis, Thies Pfeiffer, Zofia Malisz, Petra Wagner, and David Schlangen. 2012. Evaluating a minimally invasive laboratory architecture for recording multimodal conversational data. In *Proceedings of the Interdisciplinary Workshop on Feedback Behaviors in Dialog, Interspeech Satellite Workshop*, pages 39–42.

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 and Language*, 34(1):232–255.

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 K Tanenhaus and Michael J Spivey-Knowlton. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632.

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 *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue. 2015.*, pages 159–161. ACL – Association for Computational Linguistics, sep.