Simple Timing for Probabilistic Multiparty Dialogue Management

Casey Kennington
Bielefeld University
Universitätstraße 25
Bielefeld, Germany
c.kennington@cit-ec.uni-bielefeld.de

Kotaro Funakoshi
Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako
Saitama, Japan
funakoshi@jp.honda-ri.com

Yuki Takahashi
Waseda University
1-104 Totsukamachi
Shinjuku, Tokyo, Japan
y_takahashi@sugano.mech.waseda.ac.jp

Mikio Nakano
nakano@jp.honda-ri.com

ABSTRACT
We present ongoing work in multi-modal, multi-user dialogue management. We have developed a dialogue manager that can handle multiple human users for a game master robot and can simultaneously perceive its environment and perform actions. In this paper, we compare two different systems with a single user, one with this baseline functionality, and a timing-proactive system that can perform actions that we consider to be important to timing when designing dialogue systems. We explain our model and the task, and evaluate the two systems using an overhearer evaluation. We found that it is not enough to have timing-sensitive behaviour if language understanding and production are not perceived as natural.

1. INTRODUCTION
An essential aspect of dialogue between two or more participants is certainly the content, that is, what is uttered, but also the timing, or when to perform an utterance. There is a continual give-and-take with minimal overlap, yet minimal silence, making timing of utterances an intricate and important part of successful dialogue. A robust dialogue system (DS) would be amenable to timing; indeed timing would of necessity be an integral aspect.

Timing in DS often focuses on turn-taking; when a dialogue participant produces an utterance. For a standard DS, that is, in a non-situated environment (i.e., a phone conversation), a very simple notion of turn-taking follows some very big assumptions, namely, that the DS is mostly reactive to complete utterances made by a human; they follow a sequential turn-taking approach where a DS waits for silence to begin its utterance and is not sensitive to barge-in actions made by the user (i.e., it continues an utterance until complete; see discussion in [1]). These types of “ping-pong” systems are famously frustrating to human users.

Recent research has focused on incremental methods of dialogue processing that continuously perceive the state of the dialogue, and continuously determine what to output (and when) which allows the interaction to be more natural to human users. As a result, incremental systems have a more realistic notion of timing than non-incremental systems (see [5]). During comprehension, an incremental DS must try to understand as much as possible as early as possible, but must be sensitive to changes in the input (i.e., a sudden repeat-request or gesture). For production, (i.e., speech synthesis) the timing is perhaps more difficult because when to produce an utterance, or a filled pause, determines how the utterance is understood by the listeners. This is even more important when considering not just dialogue between a DS and a single human, but multi-logue between a DS and multiple humans, which is the goal of our system.

In this paper, we present a simple model of multi-user dialogue management that is what we call timing-proactive in that it not only reacts to changes in a user state, but it is ready to act at all times, even while the system is performing an utterance (such as barge-in attempts or repeat requests by humans), and handles dialogue states where a certain amount of time passes without any action by human users. Most of our system was built out of existing components (see below). We evaluate our approach using an overhearer evaluation.

At the offset we wish to note that isolating timing for evaluation in DS is a non-trivial task; here we attempt to define components of our DS that we considered to be more sensitive to timing than the baseline model.
Paper plan: we begin with a section on related work, including multi-user dialogue management and incremental processing. We then describe our model, focusing attention to the area that is novel: the Action Manager and explain how it implicitly handles actions that are sensitive to timing beyond that of simple turn-taking. We then describe our newly collected data, and the procedure for training our models. We then explain how we evaluated our approach and conclude.

2. RELATED WORK

The main focus here of our DS is the dialogue manager. Dialogue management (DM) is an active area of ongoing research, yet much of the research has focused on non-situated, two-party DM. Multiparty dialogue (multilogue) management (MDM) has been an area of interest for a number of years: [7] explored the characteristics of multilogue in comparison to two-party dialogue. There is also recent multilogue research using virtual agents [15], including aspects of turn-taking [3]. [8] give results on a MDM system for a robot bar-tender, which engaged multiple participants. Furthermore, MDM differs from standard DM because of necessary addressee identification; a good comparison of methods of which can be found in [12]).

In terms of timing for our approach, one must look at incremental dialogue processing, a good overview of which can be found in [13]. Incremental systems have been found to be more natural in their interaction than non-incremental approaches [5]. Incremental systems continuously perceive and produce, something that is very important in MDM because humans are more likely to barge-in at any time (as in [6]). Other recent work includes [9, 15] which focused on incremental language understanding, and [2] looked at incremental speech production; how systems that produce filled-pauses are preferred to systems that produce speech without updating their information state while an utterance is ongoing.

Other work has looked into turn-taking strategies [3] (i.e., determining who has the floor), though we do not explicitly focus on that here. In fact, we make an important simplifying assumption that the DS never really has the floor; it is more of a guide to help the user through the task and is allowed to speak when it decides to do so, but suspends an utterance when a user takes the floor. Other work that has looked specifically into timing is [11], which focused on the timing of feedback using a data-driven model, though we do not consider feedback here.

3. THE MODEL

We take a simplified approach to MDM and treat it as a set of individual DMS (see [14] which discusses several issues and approaches of MDM). For each DM, we apply the probabilistic rules approach as described in [10].1 This approach uses Bayesian Networks (BN) defined by a set of rules that determine the (partitioned) dialogue states and policies. The possible states need to be explicitly defined in the rules, but the BN probabilities and decision utilities can be learned from data.

1Indeed, we are using opendial developed by the author of [10]: https://code.google.com/p/opendial/

Figure 1 gives an overview of our model. We model MDM by using a single DM for each detected participant which maintain the user state from the input (sensors or annotations). These individual DMS work independently (to a certain extent; modules allow certain information to be shared). Each DM makes decisions, working in parallel, on how to respond to their corresponding users. Those action decisions are passed to an additional module called the action manager (AM; though different from the AM in [5]), which makes the final decision as to what gets realised (uttered), and when.

The part of our model that is potentially novel, and the part that implicitly handles the timing of actions, is the AM, which determines which tasks to perform and when to perform them.2 Our approach to the AM is realised as a single priority blocking queue. Though the queue structure may be familiar to the reader, we will describe the aspects of this particular type of queue and how actions of MDM and timing are implicitly handled with such a structure.

A queue is a simple first-in, first-out structure. Each DM can put actions (here, an action is defined as a DS DA and corresponding speech) onto the queue, which are read off the queue in the order that they were inserted. This forces the actions to take place sequentially, which works well in practice for MDM because of an assumption of rationality: the human users understand how to engage in normal dialogue, so the MDM can respond to each individual in order. Very simple timing information can accompany the actions, for example, how many milliseconds to wait after the end of the utterance before moving onto the next one on the queue (e.g., if the utterance is long, it might allow more time for a potential repeat request made by a user).

It is the often the case that the queue is empty, indeed the MDM should not always attempt to perform an action. This

2We say potentially novel because it is such a simple approach that someone else must have thought of it already, we just have not yet found a reference
being the case, queue has the added ability to perform blocking, simply, if the queue is empty, it suspends until there is an action in the queue. Thus, the reading of the queue is a separate process and can continue independently from perception, allowing perception to continue even in the middle of DS utterances. Our baseline system has this functionality.

With the addition of priority to the queue, we are able to implicitly handle more actions that require special timing. A priority queue allows a specific ordering of objects on the queue. If a DM (or separate module) puts an action onto the queue with a high priority, it is acted upon before other lower-priority actions, even if they were added to the queue first. This allows our MDM to easily perform repeat-requests; e.g., if someone asks the system to repeat the last utterance, the DM for that user can re-add the most recent utterance to the queue with high priority, causing it to be uttered before whatever else might already be on the queue. Figure 2 shows this graphically. To make the model proactive, a separate module offered suggestions after a defined amount of silence. This module simply put an action onto the queue which would immediately be realised.

Finally, the AM can perform incremental speech synthesis, as described in [4] (though we don’t use their functionality of augmenting or changing an utterance before it is completely realised). This allows us to handle barge-in attempts by stopping the ongoing audio and either resuming later (i.e., a filled pause), or starting the utterance over from the beginning. Furthermore if a DM puts several actions onto the queue (e.g., to explain an aspect of the task), a user can interrupt during the first action, explaining that she already understands the task, cutting off the current utterance, and removing the other utterances from the queue.

4. DATA AND PROCEDURE

Data. The corpus that was used for this project was the Twenty-Doors corpus, recently collected by Honda Research Institute. It was done in Wizard-of-Oz fashion where a human controlled a NAO robot, which interfaced with the participants. The robot (wizard) was able to see the scene through a camera and respond to the situation. When humans (potential participants) entered the scene, the robot would call out to them and invite them to play the game. If they agreed, introductions were exchanged and the rules were explained. After a confirmation of rule understanding, the game proceeded. The game itself was a 20-questions scenario, where the robot thought of a simple object (i.e., a fruit or an animal) and the participants were to ask yes/no questions in order to guess the object. The robot could respond yes, no, I don’t know, or signal some kind of misunderstanding. If the correct object was identified, the robot congratulated them on guessing correctly and attempted to play another game with them. Participants could enter or leave the scene at any time and up to 3 participants were allowed to be in the game. The NAO spoke only English, while the participants spoke a mixture of English and Japanese. The wizard could understand both English and Japanese.

An example scene is represented in Figure 3. An example dialogue is as follows:

- NAO: Wait! Do you want to play a game?
- HUMAN: Okay.
- NAO: These are the rules. First, you imagine a thing, so you guess what it is ... (further explanation)
- NAO: Do you understand?
- HUMAN: Yes.
- NAO: Okay. Ask me a yes/no question!
- HUMAN: is it red?
- NAO: no
- NAO: is it a dog?
- HUMAN: is it a dog?
- NAO: You’re right! Congratulations! Let’s play again!

A total of 23 sessions were collected and annotated by professional annotators. Annotations included (for each participant and the wizard), the transcribed speech, gaze, to whom a speaker was speaking, participation state (passing, observing, or participating), and other dialogue-specific state
variables. For this paper, because of the progress of annotations, we were able to use the first 9 sessions for development and training the models, and session 10 for evaluation.

As we are using annotated data, we are assuming fully-observed user states, though much of the recent DM research focuses on handling partially observed user states (a very important topic; there has been a recent challenge on user state tracking, see [16]). However, we have developed our model to potentially handle partially observed input in the future (which opendial can accommodate).

Procedure. The probabilistic rules for opendial that were used for the individual DMs were determined from the training data; all individual user states / wizard action pairs were considered. Each rule consisted of a set of user state variables and their values, as well as all of the possible corresponding outcomes (wizard decisions) that were observed, given that user state. Training the BN probabilities and policies (utility values of rule outcomes) was done using the same data; the training procedure uses Bayesian learning, where all values are initialized to a uniform distribution and, as training progresses, it acts as a simulator that learns from observed user state and wizard decision pairs, then adjusts the distributions accordingly (see [10] for a more detailed explanation; training was done using opendial). To handle states that appear during evaluation, but were not observed during training, we applied a simple back-off approach: opendial considers rules in order, so the rules were ordered from most strict (rules that considered the most variables) to least strict (rules with only one or two variables), so even if a certain user state is reached during evaluation, the model can make a decision even if it that means it has to ignore some of the user state variables.

Below is an example of a rule in opendial format that considers four state variables and provides two possible actions. In this state, the user is returning a greeting, but still only observing the scene (not yet participating). The effect utility value is replaced with an estimated value after training. This example also shows that the speech produced by our system is a set of canned responses, which can be replaced by a language generation component.

```
<case>
  <if var="DA" value="Return-Greeting"/>
  <if var="Phase" value="Engagement"/>
  <if var="Speaking" value="toNAO"/>
  <if var="Participating" value="Observing"/>
</case>
<condition>
  <effect util="theta_decision1">
    <set var="dialogue-act" value="Suggestion"/>
    <set var="ds-speech" value="let’s play the game"/>
  </effect>
</condition>
<effect util="theta_decision2">
  <set var="dialogue-act" value="Introduce"/>
  <set var="ds-speech" value="my name is nao"/>
</effect>
</case>
```

The input state was fully observed and fed directly from our annotated corpus into Opendial DM which determined what to speak, and when. Speech was generated using an incre-mental version of Mary TTS [2], which allowed the functionality of pauses and restarts.

5. OVERHEARER EVALUATION

![Diagram](image)

Figure 4: Evaluation system differences; baseline system could perceive and produce simultaneously; the timing-proactive system, which is equipped with the additional modules represented in the dashed box, has the added ability to handle barge-ins (including repeat requests), give hints during inactivity, and pause or resume the system speech (i.e., produce filled pauses).

Similar to [5], we used a simple overhearer evaluation: humans were to compare two dialogue systems. Figure 4 shows the difference between the two systems. The baseline system had the basic functionality of the AM queue in that it could perceive input and produce simultaneously, but was strictly reactive to user input (not proactive). The timing-proactive version had mechanisms for handling timing-sensitive states, such as (1) barge-in attempts (e.g., mid-sentence repeat requests), (2) time-sensitive spacing between actions, and (3) making suggestions after a certain amount of time of inactivity. Because of the nature of the evaluation (session 10 of our data), our evaluation focused on the timing-proactivity of interaction with a single user (that is, multiparty aspects of our model were not evaluated here, but left for future work). A total of 11 people of varying ages, all employed by Honda Research Institute, participated in the evaluation. All but two participants were native Japanese speakers, the two others were native English speakers. All participants could speak Japanese and English.

The participants were to observe both DS types (baseline and timing-proactive) and evaluate each independently. The participants were split into two groups, each group observed the two systems in a different order. The participants were asked to observe the first system then fill out a questionnaire, then observe the second system and fill out the same questionnaire for that system. To assure that the two groups observed the same dialogues, the dialogues were presented as pre-recorded videos of the two DS types. For each type, session 10 of our data was used; the human utterances were the same in both versions, but the NAO utterances were replaced by each of our dialogue systems.

Three questions were asked and participants could indicate their agreement using a seven-point Likert scale (1 being complete disagreement, 7 being complete agreement). The questions were asked after each system was presented and were as follows:
• Q1. the system sounded natural to me
• Q2. the system was responsive
• Q3. the system was proactive

Results. Table 1 shows the mean scores and standard deviations over all 11 participants for each question, for each system. Q1 was significant (t=0.02, p<0.05) where the baseline was preferred, but Q1 and Q3 were not (respectively, t=0.69, t=0.08). Though the timing-proactive system was clearly perceived as more proactive, it was not perceived as more natural than the baseline system.

<table>
<thead>
<tr>
<th></th>
<th>baseline</th>
<th>timing-proactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>3.7 (1.61)</td>
<td>2.63 (1.03)</td>
</tr>
<tr>
<td>Q2</td>
<td>3.8 (1.32)</td>
<td>3.6 (1.62)</td>
</tr>
<tr>
<td>Q3</td>
<td>3.6 (1.28)</td>
<td>4.5 (1.13)</td>
</tr>
</tbody>
</table>

Table 1: Means (and std. dev.) of the overhearer evaluation questions for Q1 (naturalness), Q2 (responsiveness) and Q3 (proactivity)

Discussion. These results are unexpected; it was our intuition that timing-proactive system would be perceived as more natural and more responsive. Some useful comments from the participants could help determine why that was not the case: it was noted that though the timing-proactive system exhibited behavior to make it pause during a barge-in attempt by the user, it did not update its utterance; it either later completed what it had already started uttering, or it started the utterance over from the beginning (as found in [2]). There were other distractions, such as the speech synthesis not sounding "human enough" and, during the introduction phase of the dialogue, some system utterances were repeated several times (e.g., my name is nao was uttered when a Greeting/Introduction dialogue act was detected, which happened 6 times in a row), something that would benefit from better language understanding beyond that of simple dialogue act recognition.

In short (in terms of naturalness and responsiveness), improvement in timing is not enough if the language understanding and production are too distracting. Certainly this is the case as the perception of naturalness would of necessity include competence in language. Our system was competent to a certain degree, but as is the case with many dialogue systems, it lacked in language understanding and production (as our focus here was on the DM). Also, our baseline system already exhibited behaviour beyond traditional DS in that it perceives input and produces output simultaneously, and that is enough to be perceived as natural and responsive, even if it lacks simple incremental functionality. This would need to be shown, however, and is left for future work.

6. CONCLUSION
In the future, we anticipate that our individual DMs will be trained on larger amounts of data, providing more coverage and better policy estimates, the incorporation of a natural language understanding component and improved speech synthesis for improved naturalness, and further evaluations using multiple-users. We antipicate that our system will be used as a DS for a fully-automated master game robot that can be used for the Twenty-Doors task and interact with humans in real-time.

Acknowledgements. Thank you to Michiru Nishbe for annotations, those who participated in the evaluation, and the anonymous reviewer for the useful comments.

7. REFERENCES


