A Temporal Simulator for Developing Turn-Taking Methods for Spoken Dialogue Systems

Ethan O. Selfridge and Peter A. Heeman
Center for Spoken Language Understanding
Oregon Health & Science University
20000 NW Walker Rd., Beaverton, OR, 97006
selfridg@ohsu.edu, heemanp@ohsu.edu

Abstract

Developing sophisticated turn-taking behavior is necessary for next-generation dialogue systems. However, incorporating real users into the development cycle is expensive and current simulation techniques are inadequate. As a foundation for advancing turn-taking behavior, we present a temporal simulator that models the interaction between the user and the system, including speech, voice activity detection, and incremental speech recognition. We describe the details of the simulator and demonstrate it on a sample domain.

1 Introduction and Background

Effective turn-taking is critical for successful human-computer interaction. Recently, approaches have been proposed to improve system turn-taking behavior that use reinforcement learning (Jonsdottir et al., 2008; Selfridge and Heeman, 2010), decision theory (e.g., Raux and Eskenazi, 2009), and hard-coded policies (e.g., Skantze and Schlangen , 2009). Some of these methods model turn-taking as content-free decisions (Jonsdottir et al., 2008; Skantze and Schlangen, 2009), while others primarily rely on dialogue context (Selfridge and Heeman, 2010) and lexical cues (e.g., Raux and Eskenazi, 2009). Turn-taking continues to be an area of active research and its development is vital for next-generation dialogue systems, especially as they allow for more mixed initiative interaction.

Researchers have turned to simulation since developing a dialogue system with real users is expensive, time consuming, and sometimes impossible. Some turn-taking simulations have been highly stylized and only model utterance content, failing to give a realistic model of timing (Selfridge and Heeman, 2010). Others have modeled a content-free form of turn-taking and *only* attend to timing and prosodic information (Jonsdottir et al., 2008; Baumann, 2008; Padilha and Carletta, 2002). The former is insufficient for the training of deployable real-time systems, and the latter neglect an important aspect of turn-taking: semantic information (Gravano and Hirschberg, 2011).

The overall goal is to develop a simulation environment to train behavior policies that can be transferred with minimal modifications to production systems. This paper presents some first steps towards this goal. We describe a temporal simulator that models the timing and content of both user and system speech, as well as that of incremental speech recognition (ISR) and voice activity detection (VAD). We detail the overall temporal simulator architecture, the design of the individual agents that simulate dialogue, and an instantiation of a simple domain. To demonstrate the utility of the simulator, we implement multiple turn-taking polices and use it to compare these policies under conditions of varying reaction time and speech recognition accuracy.

2 Temporal Simulation Framework

We now describe the details of the temporal simulator. Inspired by the Open Agent Architecture (Martin et al., 1999), it is composed of a number of *agents*, each running as a separate computer process. We first describe the time keeping procedure and then the overall agent communication structure.

Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), pages 113–117,
Seoul, South Korea, 5-6 July 2012.

Time Keeping: To provide a useful training environment, the simulator must realistically model, and run much faster than, 'real-time'. To do this, the simulator keeps an internal clock that advances to the next time slice when all agents have been run for the current time slice. This structure allows the simulator to run far faster than 'real-time' while supporting realistic communication. This framework is similar to the clock cycle described by Padilha et al (2002).

Agent Communication: Agents use messages to communicate. Messages have three components: the addressee, the content and a time stamp. Time stamps dictate when the content is to be processed and must always be for a future, not the current, time slice, as the alternative would imply instantaneous communication and overly complicate the software architecture. A central hub receives all messages and passes them to the intended recipient agent at the appropriate time. At every slice, each agent runs two procedures: one that retrieves messages and one that can send messages. If there are multiple messages intended for the same time slice, the agent completely processes one before moving to the next.

3 Dialogue Simulator

We use the above temporal simulator to simulate dialogue. At present, we focus on dyadic interaction and have three agents that are run in a strict order at every time slice: User, ISR, and System. Time slices are modeled as 10 millisecond (ms) increments, as this is the time scale that speech recognizers run at.

In general, the User agent sends messages to the ISR agent that sends messages to the System agent. The System agent generally sends messages to both the User agent and the ISR agent. The behavior of all three agents rely on parameters (Table 1) that may either be set by hand or estimated from data. The User and System agents have near identical construction, the primary difference being that the System can misunderstand User speech.

User and System Design: Agent speech is governed by a number of timing parameters. The *Take-Turn* parameter specifies when the agent will begin speaking the selected utterance. The agent gets the first word of the utterance, sets the *Word Length* parameter, and "begins" to speak by sending a speech event message. The agent outputs the word after the specified *Word Length*, and sets the *Inter-Word Pause* parameter that governs when the next word will begin. When the agent completes the utterance, it waits until a future time slice to start another (as governed by the *Inter-Utterance Pause* parameter). However, if the listening agent interrupts mid-utterance, the speaking agent stops speaking and will not complete the utterance. Any dialogue agent architecture can be used, providing the input and output fit with the above specifications.

ISR Design: The ISR agent works as both an Incremental Speech Recognizer and a VAD. We currently model uncertainty in recognition but not in the VAD, though this is certainly a plausible and worthwhile addition. When the ISR agent receives the speech event from the User, it sets the VAD *Speech Start* parameter that models lag in speech detection, and the *Speech End (no word)* parameter that models situations where the user starts speaking but stops mid-word and produces an unrecognizable sound. When the word is received from the User, the *Speech End (word)* parameter is set and a partial phrase result is generated based on the probability that the word will be correctly recognized. This probability is then used as the basis for a confidence score that is packaged with the partial phrase result. A *Recognition Lag* parameter governs the time between User speech and the output of partial phrase results to the System. The form of ISR we model recognizes words cumulatively (see Figure 1 for an example) though the confidence score, at present, is only for the newly recognized word. The recognizer will continue to output partials from User words until the User stops speaking or the System sends a message to stop recognizing. One critical aspect of ISR which we are *not* modeling is partial instability, where partials are revised as recognition progresses. Partial instability is an area of active research (e.g. Baumann et al. 2009) and, while revisions may certainly be modeled in the future, we chose not to for simplicity's sake. We feel that, at present, the *Recognition Lag* parameter is sufficient to model the time for a partial to become stable.

Table 1: Parameters and demonstration values (ms)

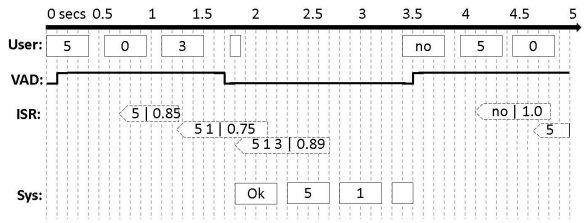
Conversant Agents	
Inter-Word pause (Usr)	$\mu = 200, \sigma = 100$
Inter-Word pause (Sys)	100
Inter-Utt. pause	$\mu = 1000, \sigma = 500$
Word Length	400
Take-Turn (Usr)	500/200
Take-Turn (Sys)	750/100
ISR Agent	
Recog. Acc.	variable
Recog. Lag	300
VAD	
Speech Start	100
Speech End (word)	200
Speech End (no word)	600

4 Simulation demonstration

We now demonstrate the utility of the temporal simulator by showing that it can be used to evaluate different turn-taking strategies under conditions of varying ASR accuracy. This is the first step before using it to train policies for use in a live dialogue system.

For this demonstration the conversant agents, the System and User, are built according to the Information State Update approach (Larsson and Traum, 2000), and perform an update for every message associated with the current time slice. The conversant agents are identical except for individual rule sets. Four types of rule sets are common across conversant agents: UNDERSTANDING rules, that update the IS using raw message content; DELIBERATION rules, that update the IS by comparing new information to old; UTTERANCE rules, that select the next utterance based on dialogue context; and TURN rules, that select the time to begin the new utterance by modifying the *Take Turn* parameter. Rule sets are executed in this order with one exception. After the UNDERSTANDING rules, the System agent has ACCEPTANCE rules that use confidence scores to decide whether to understand the recognition or not.

Figure 1: Sample dialogue with timing information

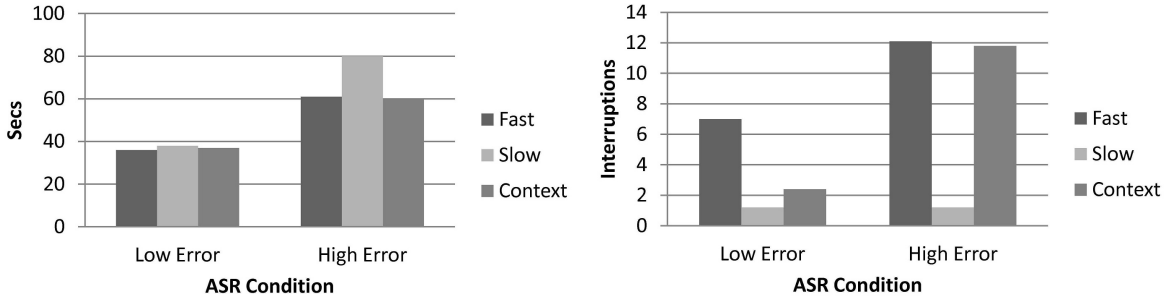


Temporal Simulation Example: We constructed a simple credit card domain, similar to Skantze and Schlangen (2009), where the User says four utterances of four digits each. The System must implicitly confirm every number and if it is correct, the User continues.[1] It can theoretically do this at any time, immediately after the word is recognized, after an utterance, or after multiple utterances. If the system says a wrong number the User interrupts the System with a "no" and begins the utterance again. The System has a Non-Understanding (NU) confidence score threshold set at 0.5. After an NU, the System will not understand any more words and will either confirm any digits recognized before the NU or, if there are no words to confirm, will say an NU utterance ("pardon?"). The User says "yes" to the final, correct confirmation. To maintain simplicity, "yes" and "no" are always accurate. If this were not the case, there would be a number of dialogues that were not successful. The User takes the turn in two ways. It either waits 500 ms after a System utterance to speak or interrupts 200 ms after the System confirms an misrecognized word, which is in line with human reaction time (Fry, 1975).

We implemented three different turn-taking strategies: two *Fixed* and one *Context-based*. Using the Fixed strategy the System either uses a *Slow* policy, waiting 750 ms after no user speech is detected, or a *Fast* policy, waiting only 100 ms. The Fast reaction time results in the System interrupting the User during an utterance when the inter-word pause becomes longer than 200 ms. This is because the VAD *Speech End* parameter is 100 ms and the System is waiting for 100 ms of silence *after Speech End*. The Slow reaction time results in far less interruptions. The Context-based turn-taking strategy uses the recognition score to choose its turn-taking behavior. The motivation is that one would want

[1] Unlike an explicit confirmation ("I heard five. Is that right?"), an implicit confirm ("Ok, five") does not necessitate a strict "yes" or "no" response.

Figure 2: Mean Time and Interruption for different turn-taking polices and ASR accuracy conditions



to confirm low-confidence recognitions sooner than those with high confidence. If any unconfirmed result has scores less than 0.8 then the System uses the Fast reaction time to try to confirm or reject as soon as possible. Alternatively, if the results all have high confidences, it can wait until a longer user pause (generally between utterances) by using the Slow reaction time. All parameter values are shown in Table 1.

Figure 1 shows a dialogue fragment of a System using the Context-based turn-taking policy. Numbers are used for the sake of brevity. The start of a box surrounding a word corresponds to when the Speech message was sent (from the User agent to the ISR agent) and the end of the box to when the word has been completed and recognition lag timer begins. The point of the ISR box refers to the time slice when the partial phrase result and score were sent to the System. Note how after the third User word the System interrupts to confirm the utterance, since the confidence score of a previous word dropped below 0.8. Also note how the User interrupts the System after it confirms a wrong number.

Comparing turn-taking policies: We evaluated the three (two Fixed and one Context-based) turn-taking policies in two conditions of ASR accuracy: Low Error, where the probability of correctness was 95%; and High Error, where the probability of correctness was 75%. We compared the mean dialogue time (left Figure 2) and the mean number of interruptions per dialogue (right Figure 2). For dialogue time, we find that all turn-taking policies perform similarly in the Low Error condition. However, in the High Error condition the Slow reaction time performs much worse since it cannot address poor recognitions with the speed of the other two. For interruption, the Fast and Context-driven policies have *far* more than the Slow for the High Error condition. However, in the Low Error condition the Fast policy interrupts far more than the Context-driven. Given that natural behavior is one goal of turn-taking, interruption, while effective at handling High Error rates, should be minimized. The Context-based policy provides support for interruption when it is needed (High Error Condition) and reduces it when it is not (Low Error Condition). The other policies are either unable to interrupt at all (Slow), increasing the dialogue time, or due to a lack the flexibility (Fast), interrupt constantly.

5 Conclusion

We take the first steps towards a simulation approach that characterizes both the content of conversant speech as well as its timing. The temporal simulator models conversant utterances, ISR, and the VAD. The simulator runs quickly (100 times faster than real-time), and is simple and highly flexible. Using an example, we demonstrated that the simulator can help understand the ramifications of different turn-taking policies. We also highlighted both the temporal nature of turn-taking — interruptions, reaction time, recognition lag...etc. — and the content of utterances — speech recognition errors, confidence scores, and wrong confirmations. Plans for future work include adding realistic prosodic modeling and estimating model parameters from data.

Acknowledgments

We thank to the reviewers for their thoughtful suggestions and critique. We acknowledge funding from the NSF under grant IIS-0713698.

References

T. Baumann, M. Atterer, and D. Schlangen. 2009. Assessing and improving the performance of speech recognition for incremental systems. In *Proc. NAACL: HLT*, pages 380–388.

T. Baumann. 2008. Simulating spoken dialogue with a focus on realistic turn-taking. In *Proc. of ESSLLI Student Session*.

D. B. Fry. 1975. Simple reaction-times to speech and non-speech stimuli. *Cortex*, 11(4):355–360.

A. Gravano and J. Hirschberg. 2011. Turn-taking cues in task-oriented dialogue. *Computer Speech & Language*, 25(3):601–634.

G.R. Jonsdottir, K.R. Thorisson, and Eric Nivel. 2008. Learning smooth, human-like turntaking in realtime dialogue. In *Proc. of IVA*, pages 162–175.

S. Larsson and D. Traum. 2000. Information state and dialogue managment in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 6:323–340.

D.L. Martin, Adam J. Cheyer, and Douglas B. Moran. 1999. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence: An International Journal*, 13(1-2):91–128.

E. Padilha and J. Carletta. 2002. A simulation of small group discussion. In *Proc. of EDILOG*, pages 117–124.

A. Raux and M. Eskenazi. 2009. A finite-state turn-taking model for spoken dialog systems. In *Proc. of HLT/NAACL*, pages 629–637.

G. Skantze and D. Schlangen . 2009. Incremental dialogue processing in a micro-domain. In *Proc. of EACL*, pages 745–753.

E.O. Selfridge and P.A. Heeman. 2010. Importance-Driven Turn-Bidding for spoken dialogue systems. In *Proc. of ACL*, pages 177–185.