An Incremental Turn-Taking Model with Active System Barge-in for Spoken Dialog Systems

Tiancheng Zhao, Alan W Black and Maxine Eskenazi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
{tianchez,awb,max+}@cs.cmu.edu

Abstract

This paper deals with an incremental turn-taking model that provides a novel solution for end-of-turn detection. It includes a flexible framework that enables active *system barge-in*. In order to accomplish this, a systematic procedure of teaching a dialog system to produce meaningful *system barge-in* is presented. This procedure improves system robustness and success rate. It includes constructing cost models and learning optimal policy using reinforcement learning. Results show that our model reduces *false cut-in* rate by 37.1% and *response delay* by 32.5% compared to the baseline system. Also the learned *system barge-in* strategy yields a 27.7% increase in average reward from user responses.

1 Introduction

Human-human conversation has flexible turn-taking behavior: back channeling, overlapping speech and smooth turn transitions. Imitating human-like turn-taking in a spoken dialog system (SDS) is challenging due to the degradation in quality of the dialog when overlapping speech is produced in the wrong place. For this, a traditional SDS often uses a simplified turn-taking model with rigid turn taking. They only respond when users have finished speaking. Thus past research has mostly focused on end-of-turn detection, finding the end of the user utterance as quickly as possible while minimizing the chance of wrongly interrupting the users. We refer here to the interruption issue as *false cut-ins* (FCs).

Recent research in incremental dialog processing promises more flexible turn-taking behavior (Atterer et al., 2008; Breslin et al., 2013). Here, the automatic speech recognizer (ASR) and natural language understanding (NLU) incrementally produce partial decoding/understating messages for decision-making. This allows for *system barge-in* (SB), starting to respond before end-of-utterance. Although this framework has shown promising results in creating flexible SDSs, the following two fundamental issues remain:

1. We need a model that unifies incremental processing and traditional turn-taking behavior.
2. We also need a systematic procedure that trains a system to produce meaningful SBs.

This paper first proposes a finite state machine (FSM) that both shows superior performance in end-of-turn detection compared to previous methods and is compatible with incremental processing. Then we propose a systematic procedure to endow a system with meaningful SB by combining the theory of optimal stopping with reinforcement learning.

Section 2 of the paper discusses related work; Section 3 describes the finite state machine; Sections 4, 5, and 6 describe how to produce meaningful SB; Section 7 gives experimental results of an evaluation using the CMU Let's Go Live system and simulation results on the Dialog State Tracking Challenging (DTSC) Corpus and Section 8 concludes.

2 Related Work and Limitations

This work is closely related to end-of-turn detection and incremental processing (IP) dialog systems.

There are several methods for detecting the end-of-turn. Raux (2008) built a decision tree for final pause duration using ASR and NLU features. At runtime, the system first dynamically chooses the final pause duration threshold based on the dialog state and then predicts end-of-turn if final pause duration is longer than that threshold. Other work explored predicting end-of-turn within a user's speech. This showed substantial improvement in speed of response (Raux and Eske-

Proceedings of the SIGDIAL 2015 Conference, pages 42–50,
Prague, Czech Republic, 2-4 September 2015.

nazi, 2009). Another approach examined prosodic and semantic features such as pitch and speaking rate in human-human conversation for turn-yielding cues (Gravano, 2009).

The key limitation of those methods is that the decision made by the end-of-turn detector is treated as a "hard" decision, obliging developers to compromise in a tradeoff between response latency and FC rate (Raux and Eskenazi, 2008). Although adding more complex prosodic and semantic features can improve the performance of the detector, it also increases computation cost and requires significant knowledge of the SDS, which can limit the accessibility for non-expert developers.

For IP, Kim (2014) has demonstrated the possibility of learning turn-taking from human dialogs using inverse reinforcement learning. Other work has focused on incremental NLU (DeVault et al., 2009), showing that the correct interpretation of users' meaning can be predicted before end-of-turn. Another topic is modeling user and system barge-in. Selfridge (2013) has presented a FSM that predicts users' barge-ins. Also, Ghigi (2014) has shown that allowing SB when users produce lengthy speech increases robustness and task success.

Different from Kim's work that learns human-like turn-taking, our approach is more related to Ghigi's method, which tries to improve dialog efficiency from a system-centric perspective. We take one step further by optimizing the turn-taking using all available features based on a global objective function with machine learning methods.

3 A Finite State Turn-Taking Model

3.1 Model Description

Our model has two distinct modes: passive and active. The passive mode exhibits traditional rigid turn-taking behavior while the active mode has the system respond in the middle of a user turn. We first describe how these two modes operate, and then show how they are compatible with existing incremental dialog approaches.

The idea is to combine an aggressive speaker with a patient listener. The speaker consists of the Text-to-Speech (TTS) and Natural Language Generation (NLG) modules. The listener is composed of the ASR and Voice Activity Detection (VAD) modules. The system attempts to respond to a user every time it detects a short pause (e.g. 100ms). But before a long pause (e.g. 1000ms) is detected, the user's continued speech will stop the system from responding, as shown on Figure 1:

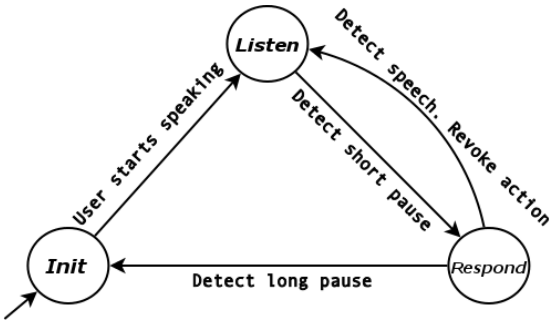

Figure 1: Turn-taking Model as a finite state machine

Most of the system's attempts to respond will thus be FCs. However, since the listener can stop the system from speaking, the FCs have no effect on the conversation (users may hear the false start of the system's prompt, but often the respond state is cancelled before the synthesized speech begins). If the attempt is correct, however, the system responds with almost 0-latency, as shown in Figure 2. Furthermore, because the dialog manager (DM) can receive partial ASR output whenever there is a short pause, this model produces relatively stable partial ASR output and supports incremental dialog processing.

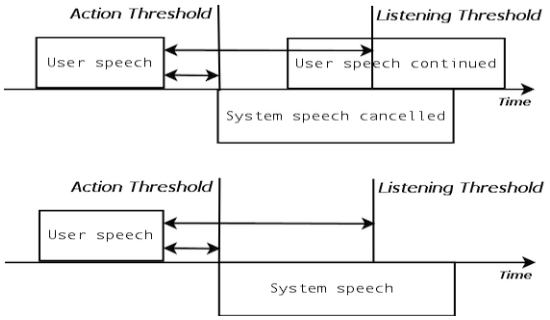

Figure 2: The first example illustrates the system canceling its response when it detects new speech before LT. The second example shows that users will not notice the waiting time between AT and LT.

We then define the short pause as the *action threshold* (AT) and the long pause as the *listening threshold* (LT), where $0 < \text{AT} \leq \text{LT}$, which can be interpreted respectively as the "aggression" and "patience" of the system. By changing the value of each of these thresholds we can modify the system's behavior from rigid turn taking to active SB.

1. Passive Agent: act fast and listen patiently (AT = small value, LT = large value)

2. Active Agent: act and listen impatiently.
   (AT = LT = small value)

This abstraction simplifies the challenge: "when the system should barge in" as the following transition: $Passive\,Agent \xrightarrow{\Phi(dialog\,state)} Active\,Agent$ where $\Phi(\cdot) : dialog\,State \rightarrow \{true, false\}$ is a function that outputs true whenever the agent should take the floor, regardless of the current state of the floor. For example, this function could output true when the current dialog states fulfill certain rules in a hand-crafted system, or could output true when the system has reached its maximal understanding of the user's intention (DeVault et al., 2009). A natural next step is to use statistical techniques to learn an optimized $\Phi(\cdot)$ based on all features related to the dialog states, in order to support more complex SB behavior.

3.2 Advantages over Past Methods

First our model solves end-of-turn detection by using a combination of VAD and TTS control, instead of trying to build a perfect classifier. This avoids the tradeoff between response latency and FC. Under the assumption that the TTS can operate at high speed, the proposed system can achieve almost 0-lag and 0-FC by setting AT to be small (e.g. 100ms). Second, the model does not require expensive prosodic and semantic turn-yielding cue detectors, thus simplifying the implementation.

4 Toward Active System Barge-in

In state-of-the-art SDS, the DM uses explicit/implicit confirmation to fill each slot and carries out an error recovery strategy for incorrectly recognized slots (Bohus and Rudnicky, 2009). The system should receive many correctly-recognized slots, thus avoiding lengthy error recovery. While a better ASR and NLU could help, Ghigh (2014) has shown that allowing the system to actively respond to users also leads to more correct slots.

Transcription	**ASR Output**
To Forbes, you know, at Squirrel Hill	To Forbes, **herron vee lyn road**
Leaving from Forbes, ⟨Noise⟩	Leaving from Forbes **from highland bus**
⟨Noise⟩, Leaving from Forbes	**PA 71C Pittsburgh, liberty** from Forbes

Table 1: Examples of wordy turns and noise presence. Bold text is the part of speech incorrectly recognized.

Table 1 demonstrates three cases where active SB can help. The first two rows show the first half of the user's speech being correctly recognized while the second half is not. In this scenario, if, in the middle of the utterance, the system can tell that the existing ASR hypothesis is sufficient and actively barges on the user, it can potentially avoid the poorly-recognized speech that follows. The third example has noise at the beginning of the user turn. The system could back channel in the middle of the utterance to ask the user to go to a quieter place or to repeat an answer. In these examples active SB can help improve robustness:

1. Barge in when the current hypothesis has high confidence and contains sufficient information to move the dialog along.
2. Barge in when the hypothesis confidence is low and the predicted future hypothesis will not get better. This can avoid recovering from a large number of incorrect slots.

A natural choice of objective function to train such a system is to maximize the expected quality of information in the users' utterances. The quality of the recognized information is positively correlated to number of correctly recognized slots (CS) and inversely correlated to the number of incorrectly recognized slots (ICS). In the next section, we describe how we transform CS and ICS into a real-value reward.

5 A Cost Model for System Barge-in

We first design a cost model that defines a reward function. This model is based on the assumption that the system will use explicit confirmation for every slot. We choose this because it is the most basic dialog strategy. A sample dialog for this strategy is as follows:

***Sys**: Where do you want to leave from?*
***User**: Leaving from X.*
***Sys**: Do you mean leaving from Y?*
***User**: No.*
***Sys**: Where do you want to leave from?*
***User**: <No Parse>*
***Sys**: Where do you want to leave from?*
***User**: I am leaving from X.*
***Sys**: Do you mean X?*
***User**: Yes.*

Given this dialog strategy the system spends one turn asking the question, and k turns confirming k slots in the user response. Also, for no-parse (0 slot) input, the system asks the same question again. Therefore, the minimum number of turns required

to acquire n slots is $2n$. However, because user responses contain ICS and no-parses, the system takes more than $2n$ turns to obtain all the slot information (assume confirmation are never misrecognized).

We denote cs_i and ics_i as the number of correctly/incorrectly recognized slots in the user response. So the quality of the user response is captured by a tuple, (cs_i, ics_i). The goal is to obtain a reward function that maps from a given user response (cs_i, ics_i) to a reward value $r_i \in \Re$. This reward value should correlate with the overall efficiency of a dialog, which is inversely correlated with the number of turns needed for task completion.

Then for a dialog task that has n slots to fill, we can denote h_i as the number of turns already spent, f_i as the estimated number of future turns needed for task completion and $E[S]$ as the expected number of turns needed to fill 1 slot. Then for each new user response (cs_i, ics_i), we update the following recursive formulas:

Initialization: $h_0 = 0, f_0 = nE[s]$
Update Rules:

$$h_i = h_{i-1} + \underbrace{1}_{\text{question}} + \underbrace{cs_i + ics_i}_{\text{confirm}} \quad (1)$$

$$f_i = f_{i-1} - \underbrace{cs_i E[S]}_{\text{acquired slots}} \quad (2)$$

Based on the above setup, it is clear that $h_i + f_i$ equals the estimated total number of turns needed to fill n slots. Then the reward, r_i, associated with each user response can be expressed as the difference between the previous and current estimates:

$$r_i = (h_{i-1} + f_{i-1}) - (h_i + f_i) \quad (3)$$

$$= -1 + \underbrace{(E[S] - 1)}_{\text{weight to CS}} cs_i - ics_i \quad (4)$$

Therefore, a positive reward means the new user response reduces the estimated number of turns for task completion while a negative reward means the opposite. Another interpretation of this reward function is that for no-parse user response ($cs_i = 0, ics_i = 0$), the cost is to waste 1 turn asking the same question again. When there is a parse, each correct slot can save $E[S]$ turns in the future, while each slot, regardless of its correctness, needs a 1-turn confirmation. As a result, this rewards function is correlated with the global efficiency of a dialog because it assigns a corpus-dependent weight to cs_i, based on $E[S]$ estimated from historical dialogs.

6 Learning Active Turn-taking Policy

After modeling the cost of a user turn, we learn a turn-taking policy that can maximize the expected reward in user turns, namely the $\Phi(dialog\ state)$ that controls the switching between passive and active agent of our FSM in Section 3.1. Before going into detail, we first introduce the optimal stopping problem and reinforcement learning.

6.1 Optimal Stopping Problem and Reinforcement Learning

The theory of optimal stopping is an area of mathematics that addresses the decision of when to take a given action based on a set of sequentially observed random variables, in order to maximize an expected payoff (Ferguson, 2012).
A formal description is as follows:

1. A sequence of random variables $X_1, X_2...$
2. A sequence of real-valued reward functions, $y_0, y_1(x_1), y_2(x_1, x_2)...$

The decider may observe the sequence $x_1, x_2...$ and after observing $X_1 = x_1, ...X_n = x_n$, the decider may stop and receive the reward $y_n(x_1, ...x_n)$, or continue and observe X_{n+1}. The optimal stopping problem searches for an optimal stopping rule that maximizes the expected reward.

Reinforcement learning models are based on the *Markov decision process* (MDP). A (finite) MDP is a tuple $(S, A, \{P_{sa}\}, \gamma, R)$, where:

- S is a finite set of N states
- $A = a_1, ...a_k$ is a set of k actions
- $P_{sa}(\cdot)$ are the state transition probabilities on taking action a in state s.
- $\gamma \in [0, 1)$ is the discount factor
- $R : S \rightarrow \Re$ is the rewards function.

Then a policy, π , is a mapping from each state, $s \in S$ and action $a \in A$, to the probability $\pi(s, a)$ of taking action a when in state s (Sutton and Barto, 1998). Then, for MDPs, the Q-function, is the expected return starting from s taking action a and thereafter following policy π and has the Bellman equation:

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a)V^\pi(s'). \quad (5)$$

The goal of reinforcement learning is to find the optimal policy π^*, such that $Q^\pi(s, a)$ can be maximized. Thus the optimal stopping problem can be formulated as an MDP, where the action space contains two actions $\{wait, stop\}$. Also, solving the optimal stopping rule is equivalent to finding the optimal policy, π^*.

6.2 Solving Active Turn-taking

Equipped with the above two frameworks, we first show that SB can be formulated as an optimal stopping problem. Then we propose a novel, non-iterative, model-free method for solving for the optimal policy.

An SDS dialog contains N user utterances. Each user utterance contains K partial hypotheses and each partial hypothesis, p_i, is associated with a tuple (cs_i, ics_i) and a feature vector, $x_i \in \Re^{f \times 1}$, where f is the dimension of the feature vector. We also assume that every user utterance is independent of every other utterance. We will call one user utterance an *episode*.

In an *episode*, the turn-taking decider will see each partial hypothesis sequentially over time, At each hypothesis it takes an action from $\{wait, stop\}$. $Wait$ means it continues to listen. $Stop$ means it takes the floor. The turn-taking decider receives 0 reward for taking the action $wait$ and receives the reward r_i from (cs_i, ics_i) according to our cost model for taking the action $stop$. This is an optimal stopping problem that can be formulated as an MDP:

- $S = \{x_1, ...\{x_1 ... x_K\}\}$
- $A = \{wait, stop\}$
- $R = -1 + (E[S] - 1)cs_i - ics_i$

Then the Bellman equations are:

$$Q^\pi(s, stop) = R(s) = r(s) \quad (6)$$

$$Q^\pi(s, wait) = \gamma \sum_{s'} P(s'|s, a)V^\pi(s') \quad (7)$$

The first equation shows that the Q-value for any state, s, with action, $stop$, is simply the immediate reward for s. The second equation shows that the Q-value for any state s, with action, $wait$, only depends on the future return by following policy π. This result is crucial because it means that $Q^\pi(s, stop)$ for any state, s, can be directly calculated based on the cost model, independent of the policy π. Also, given a policy π, $Q^\pi(s, wait)$can also be directly calculated as the discounted reward the first time that the policy chooses to stop.

Meanwhile, for a given *episode* with known reward r_i for each partial hypothesis p_i, optimal stopping means always to stop at the largest reward, meaning that we can obtain the oracle action for the training corpus. Given a sequence of reward $(r_i, ...r_K)$, the optimal policy, π, chooses to stop at partial p_m if $m = \arg\max_{j \in (i,K]} r_j$.

The Bellman equations become:

$$Q^\pi(s_i, stop) = r_i \quad (8)$$

$$Q^\pi(s_i, wait) = \gamma^{m-i} r_m \quad (9)$$

and the oracle action at any s can be obtained by :

$$a_i^* = wait \quad \text{if } Q^*(s_i, stop) < Q^*(s_i, wait)$$

$$a_i^* = stop \quad \text{if } Q^*(s_i, stop) \geq Q^*(s_i, wait)$$

This special property of optimal stopping problem allows us to use supervised learning methods directly modeling the optimal Q function, by finding a mapping from the input state space, s_i, into the Q-value for both actions: $Q(s_i, stop)^*$ and $Q(s_i, wait)^*$. Further, inspired by the work of reinforcement learning as classification (Lagoudakis and Parr, 2003), we decide to map directly from the input state space into the action space: $S \rightarrow A^*$, using a Support Vector Machine (SVM).

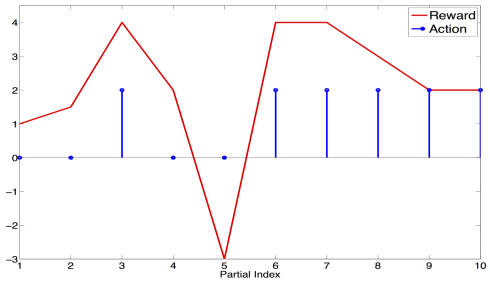

Figure 3: An example showing the oracle actions for one episode. $1 = stop$ and $0 = wait$.

Advantages of solving this problem as a classification rather than a regression include: 1) it explicitly models $sign(Q(s_i, stop)^* - Q(s_i, wait)^*)$, which sufficiently determines the behavior of the agent. 2) SVM is known as a state-of-the-art modeler for the binary classification task, due to its ability to find the separating hyperplane in nonlinear space.

6.3 Feature Construction

Since SVM requires a fixed input dimension size, while the available features will continue to increase as the turn-taking decider observes more partial hypotheses, we adopt the functional idea used by the openSMILE toolkit (Eyben et al., 2010). There are three categories of features: immediate feature, delta feature and long-term feature. Immediate features come from the ASR and the NLU in the latest partial hypothesis. Delta features are the first-order derivate of immediate features with respect to the previous observed feature. Long-term features are global statistics associated with all the observed features.

Immediate Features	
Final pause duration	Number of slots
Hypothesis stability	Transitions of (no)parse
Frame number	Number of words
Utterance duration	Number of unparsed gap
Language model score	Unparsed percentage
Word confidence	Max of pause duration
Number of noun	Mean of pause duration
Boundary LM score	Var of pause duration
First level matched	Hypothesis confidence
Long-term Functional Features	
Mean	Standard Deviation
Maximum	Position of maximum
Minimum	Position of miniumu

Table 2: List of immediate/long-term features

Table 2 shows that we have 18 immediate features, 18 delta features and $18 \times 7 = 126$ long-term features. Then we apply F-score feature selection as described in (Chen and Lin, 2006). The final feature set contains 138 features.

7 Experiments and Results

We conducted a live study and a simulation study. The live study evaluates the model's end-of-turn detection. The simulated study evaluates the active SB behavior.

7.1 Live Study

The finite state machine was implemented in the Interaction Manager of the CMU Lets Go system that provides bus information in Pittsburgh (Raux et al., 2005). We compared base system data from November 1-30, 2014 (773 dialogs), to data from our system from December 1-31, 2014 (565 dialogs).

The base system used the decision tree end-of-turn detector described in (Raux and Eskenazi, 2008) and the active SB algorithm described in (Ghigi et al., 2014). The *action threshold* (AT) in the new system was set at 60% of the decision tree output in the former system and the *listening threshold* (LT) was empirically set at $1200ms$.

7.2 Live Study Metrics

We observed that FCs result in several users' utterances having overlapping timestamps due to a built-in 500ms padding before an utterances in PocketSphinx. This means that we consider two consecutive utterances with a pause less than 500ms as one utterance. Figure 4 shows that when the end-of-turn detector produces an FC, the continued flow of user speech instantiates a new user utterance which overlaps with the previous one. In this example, utterances 0 and 1 have overlaps while utterance 2 does not. So users actually produce two utterances, while the system thinks there are three due to FC.

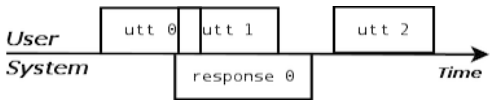

Figure 4: Utterance fragments caused by FCs. This example has $UFR = \frac{2}{3}$.

Thus, we can automatically calculate the FC rate of every dialog, by counting the number of user utterances with overlaps. We define an utterance fragment ratio (UFR) that measures the FC rate in a dialog.

$$UFR = \frac{\text{Number of user utterances with overlaps}}{\text{Total number of user utterances}}$$

We also manually label task success (TS) of all the dialogs. We define TS as: a dialog is successful if and only if the system conducted a back-end search for bus information with all required slots correctly recognized. In summary, we use the following metrics to evaluate the new system:

1. Task success rate
2. Utterance fragment ratio (UFR)
3. Average number of *system barge-in* (ANSB)
4. Proportion of long user utterances interrupted by *system barge-in* (PLUISB)
5. Average response delay (ARD)
6. Average user utterance duration over time

7.3 Live Study Results

Table 3 shows that the TS rate of the new system is 7.5% higher than the previous system (p-value $<$ 0.01). Table 4 shows that overall UFR decreased by 37.1%. UFR for successful and for failed dialogs indicates that the UFR decreases more in failed dialogs than in successful ones. One explanation is that failed dialogs usually have a noisier environment. The UFR reduction explains the increase in success rate since UFRs are positively correlated with TS rate, as reported in (Zhao and Eskenazi, 2015)

Table 5 shows that the SB algorithm was activated more often in the new system. This is because the SB algorithm described in (Ghigi et al., 2014) only activates for user utterances longer than 3 seconds. FCs will therefore hinder the ability of this algorithm to reliably measure user utterance dura-

	Success	Failed	TS Rate	P-value
New System	271	294	**48.0%**	**0.0096**
Old System	321	452	41.5%	

Table 3: Success rate between old and new systems. P-value is obtained via Wald Test

UFR	Overall	Successful dialog	Failed dialog
New System	12.2%	9.2%	15.0%
Old System	19.4%	12.5%	24.3%

Table 4: Breakdown into successful/failed dialogs

tion. This is an example of how reliable end-of-turn detection can benefit other SDS modules. Table 5 also shows that the new system is 32.5% more responsive than the old system. We purposely set the action threshold to 60% of the threshold in the old system, which demonstrates that the new model can have an response speed equals to action threshold that is independent of the FC rate.

Metric	Old System	New System
ANSB	1.04	1.50
PLUISB	53.9%	77.8%
ARD (ms)	853.49	576.09

Table 5: Comparison of barge-in activation rate and response delay

Figure 5 shows how average user utterance duration evolves in a dialog. Utterance duration is more stable in the new system than in the old one. Two possible explanations are: 1) since UFR is much higher in the old system, the system is more likely to cut in at the wrong time, possibly making users abandon their normal turn-taking behavior and talk over the system. 2) more frequent activation of the SB algorithm entrains the users to produce more concise utterances.

7.4 Simulation Study

This part of the experiment uses the DSTC corpus training2 (643 dialogs) (Black et al., 2013). The data was manually transcribed. The reported 1-best word error rate (WER) is 58.2% (Williams et al., 2013). This study focuses on all user responses to:"Where are you leaving from?" and "Where are you going?" which have 688 and 773 utterances respectively.

An automatic script, based on the manual transcription, labels the number of correct and incorrect

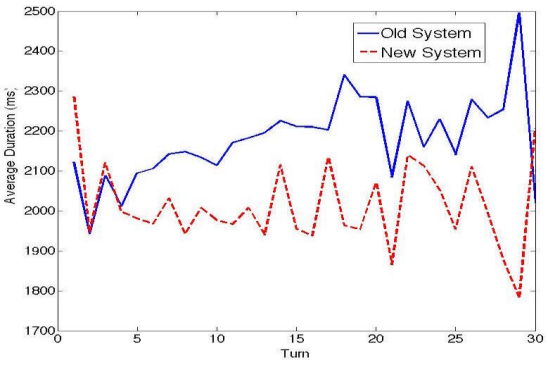

Figure 5: Average user utterance duration over the index of user turns in a dialog.

slots (cs_i, ics_i) for each partial hypothesis, p_i. Also from the training data, the expected number of turns needed to obtain 1 slot, $E[S]$, is 3.82. For simplicity, $E[S]$ is set to be 4. So the reward function discussed in Section 5 is: $r_i = -1 + 3cs_i - ics_i$.

After obtaining the reward value for each hypothesis, the oracle action at each partial hypothesis is calculated based on the procedure discussed in Section 6.3 with $\gamma = 1$.

We set the SVM kernel as RBF kernel and use a grid search to choose the best parameters for cost and kernel width using 5-fold cross validation on the training data (Hsu et al., 2003). The optimization criterion is the F-measure.

7.5 Simulation Study Metrics

The evaluation metrics have two parts: classification-related (precision and recall) and dialog-related. Dialog related metrics are:

1. Accuracy of system barge-in
2. Average decrease in utterance duration compared to no *system barge-in*
3. Percentage of no-parse utterance
4. Average CS per utterance
5. Average ICS per utterance
6. Average reward $= 1/T \sum_i r_i$, where T is the number of utterances in the test set.

The learned policy is compared to two reference systems: the oracle and the baseline system. The oracle directly follows optimal policy obtained from the ground-truth label. The baseline system always waits for the last partial (no SB).

Furthermore, a simple smoothing algorithm is applied to the SVM output for comparison. This algorithm confirms the stop action after two consecutive stop outputs from the classifier. This increases the classifier's precision.

7.6 Simulation Study Results

10-fold cross validation was conducted on the two datasets. Instead of using the SVM binary output, we apply a global threshold of 0.4 on the SVM decision function for output to achieve the best average reward. The threshold is determined based on cross-validation on training data.

Table 6 shows that the SVM classifier can achieve very high precision and high recall in predicting the correct action. The F-measure (after smoothing) is 84.46% for departure question responses and 85.99% for arrival questions.

	Precision	**Recall**	**Precision (smooth)**	**Recall (smooth)**
D	92.64%± 2.88	78.04%± 2.39	93.86%± 2.80	76.79%± 2.35
A	93.59%± 2.42	79.64%± 3.41	93.63%± 2.30	79.51%± 3.04

Table 6: Cross-validation precision and recall with standard error for SVM. D = responses to departure question, A = responses to arrival question.

Table 7 shows that learned policy increases the average reward by 27.7% and 14.9% compared to the baseline system for the departure and arrival responses respectively. We notice that the average reward of the baseline arrival responses is significantly higher. A possible reason is that by this second question the users are adapting to the system.

The decrease in average utterance duration shows some interesting results. For responses to both questions, the oracle system utterance duration is about 55% shorter than the baseline one. The learned policy is also 45% shorter, which means that at about the middle of a user utterance, the system can already predict that the user either has expressed enough information or that the ASR is so wrong that there is no point of continuing to listen.

	Depature		**Arrival**	
Policy	Average reward	Average duration decrease	Average reward	Average duration decrease
Baseline	0.795	0%	0.959	0%
Oracle	1.396	58.1%	1.430	55.7%
Learned	0.998	42.8%	1.089	47.6%
Learned (smooth)	**1.016**	45.6%	**1.102**	46.2%

Table 7: Average reward and duration decrease for baseline, oracle, SVM and smooth SVM system.

Table 8 expands our understanding of the oracle and learned policy behaviors. We see that the oracle produces a much higher percentage of no-parse utterances in order to maximize the average reward, which, at first, seems counter-intuitive. The reason is that some utterances contain a large number of incorrect slots at the end and the oracle chooses to barge in at the beginning of the utterance to avoid the large negative reward for waiting until the end. This is the expected behavior discussed in Section 4. The learned policy is more conservative in producing no-parse utterances because it cannot cheat like the oracle to access future information and know that all future hypotheses will contain only incorrect information. However, although the learned policy only has access to historical information, it manages to predict future return by increasing CS and reducing ICS compared to the baseline.

Policy	**No-parse percent**	**Average CS**	**Average ICS**
Baseline	6.86%	0.765	0.499
Oracle	14.71%	0.865	0.196
Learned	8.14%	**0.796**	0.389
Learned (smooth)	**8.71%**	0.789	**0.360**

Table 8: No parse percentages and average CS and ICS for responses to the departure question.

8 Conclusions and Future Directions

This paper describes a novel turn-taking model that unifies the traditional rigid turn-taking model with incremental dialog processing. It also illustrates a systematic procedure of constructing a cost model and teaching a dialog system to actively grab the conversation floor in order to improve system robustness. The turn-taking model was tested for end-of-turn detection and active SB. The proposed model has shown superior performance in reducing FC rate and *response delay*. Also, the proposed SB algorithm has shown promise in increasing the average reward in user responses.

Future studies will include constructing a more comprehensive cost model that not only takes into account of CS/ICS, but also includes other factors such as conversational behavior. Further, since $E[S]$ will decrease after applying the learned policy, it invalidates the previous reward function. Future work should investigate how the change in $E[S]$ impacts the optimality of the policy. Also, we will add more complex actions to the system such as back channeling, clarifications etc.

References

Michaela Atterer, Timo Baumann, and David Schlangen. 2008. Towards incremental end-of-utterance detection in dialogue systems. *Proceedings of the 22nd International Conference on Computational Linguistics*.

Alan Black, Maxine Eskenazi, Milica Gasic, Helen Hastie, KAIST Kee-Eung Kim, Korea Ian Lane, Sungjin Lee, NICT Teruhisa Misu, Japan Olivier Pietquin, France SUPELEC, et al. 2013. Dialog state tracking challenge. http://research.microsoft.com/en-us/events/dstc/.

Dan Bohus and Alexander I Rudnicky. 2009. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361.

Catherine Breslin, Milica Gasic, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. Continuous asr for flexible incremental dialogue. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8362–8366. IEEE.

Yi-Wei Chen and Chih-Jen Lin. 2006. Combining svms with various feature selection strategies. In *Feature extraction*, pages 315–324. Springer, Berlin Heidelberg.

David DeVault, Kenji Sagae, and David Traum. 2009. Can i finish?: learning when to respond to incremental interpretation results in interactive dialogue. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 11–20. Association for Computational Linguistics.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the international conference on Multimedia*, pages 1459–1462. ACM.

Thomas S Ferguson. 2012. *Optimal stopping and applications*. University of California, Los Angeles.

Fabrizio Ghigi, Maxine Eskenazi, M Ines Torres, and Sungjin Lee. 2014. Incremental dialog processing in a task-oriented dialog. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Agustin Gravano. 2009. *Turn-taking and affirmative cue words in task-oriented dialogue*. Ph.D. thesis.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University.

Dongho Kim, Catherine Breslin, Pirros Tsiakoulis, Milica Gašic, Matthew Henderson, and Steve Young. 2014. Inverse reinforcement learning for micro-turn management. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 328–332. International Speech and Communication Association.

Michail Lagoudakis and Ronald Parr. 2003. Reinforcement learning as classification: Leveraging modern classifiers. In *ICML*, volume 3, pages 424–431.

Antoine Raux and Maxine Eskenazi. 2008. Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 1–10. Association for Computational Linguistics.

Antoine Raux and Maxine Eskenazi. 2009. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 629–637. Association for Computational Linguistics.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Lets go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005*.

Ethan Selfridge, Iker Arizmendi, Peter Heeman, and Jason Williams. 2013. Continuously predicting and processing barge-in during a live spoken dialogue task. In *Proceedings of the SIGDIAL 2013 Conference*.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. MIT Press.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Tiancheng Zhao and Maxine Eskenazi. 2015. Human-system turn taking analysis for the let's go bus information system. Pittsburgh, May. The Meeting of the Acoustical Society of America.