interspeech 2008 incorporating SST 08

Dialog Management using Weighted Finite-State Transducers

Chiori Hori [†‡], *Kiyonori Ohtake* [†‡], *Teruhisa Misu* [†‡],
Hideki Kashioka [†‡], *Satoshi Nakamura* [†‡]

[†] National Institute of Information and Communications Technology (NICT)
[‡] Advanced Telecommunications Research Institute International (ATR)
Keihanna Science City, Kyoto, 619-0288, Japan
{chiori.hori@atr.jp}

Abstract

We are aiming to construct an expandable and adaptable dialog system which handles multiple tasks and senses users' intention via multiple modalities. A flexible platform to integrate different dialog strategies and modalities is indispensable for this purpose. In this paper, we propose an efficient approach to manage a dialog system using a weighted finite-state transducer (WFST) in which users' concept and system's action tags are input and output of the transducer, respectively. By incorporating WFSTs in dialog management, different components can easily be integrated and work on a common platform. We have constructed a prototype spoken dialog system of the Kyoto tour guide which assists users in making a plan for one-day trip to sightsee through interaction. A WFST for dialog management was created based on the annotated transcript of the Kyoto tour guide dialog corpus we recorded. The WFST was then composed with a word-to-concept WFST for language understanding, and optimized. We have confirmed our WFST-based dialog manager accepted recognition results from a speech recognizer well and worked as we designed.

Index Terms: Kyoto Tour Guide Dialog corpus, spoken dialog, weighted finite-state transducer (WFST), dialog management, concept tag, spoken language understanding

1. Introduction

Dialog systems help users accomplish a task through (spoken or multi-modal) interaction with machines [1]. When a user requests something by speaking and/or pointing to a system, the system responds to the user's request. To construct a dialog system, it is necessary to manually or semi-automatically design a scenario which handles dialog in response to user' input so as to accomplish a task efficiently. It also needs human-machine interfaces such as a speech recognizer, a spoken language understanding unit, a gesture recognizer, a speech synthesizer, etc., which should be combined appropriately according to situations it works in.

Our goal is constructing a human-machine dialog system which enables users to interact with machines using spontaneous interaction via multiple modalities. Such a dialog system requires functions to let users behave spontaneously to indicate their intentions to the system, understand users' intentions represented by verbal and nonverbal expressions, and proactively handle dialog using all sensed signals. The state-of-the-art technologies for human-machine dialog are not sufficient to let humans spontaneously speak to systems. Even though a system seems to behave like a human, the system cannot accept humans' spontaneous reactions well.

To design a complex scenario which absorbs users' spontaneity, we may need to combine several scenarios written in different fashions such as finite-state automaton, frame-based representation, if-then rules, etc. We can also take stochastic approaches such as Markov decision process (MDP) [2], partially observable MDP (POMDP) [3] when the model can be trained appropriately with enough data. However, it is not easy to control different fashions of scenarios in a dialog system considering multi-modal inputs, and therefore enormous labor is required to implement such functions. Accordingly, we need an expandable and adaptable integration platform which enables us to separately design several scenarios and functions to handle system actions in response to user's input.

This paper proposes an efficient approach to organize a dialog system using weighted finite-state transducers (WFSTs) in which dialog scenarios are represented in WFST form. Although WFSTs are mainly used in speech and language processing [4], we use them for dialog management where input symbols of the WFST are concept tags indicating user's intention while its output symbols are action tags indicating system's actions. Concept tags of user's input are translated into the corresponding system actions by the WFST.

Since the WFST framework provides us a general representation, many types of scenarios can be converted into WFSTs. Once a scenario is represented in a WFST, it can be combined with other WFSTs and driven with our WFST-based dialog manager. Furthermore, additional knowledge can be easily incorporated in scenario WFSTs. For example, after a human draw a non-deterministic WFST as a scenario, n-gram probabilities of tags in a dialog corpus can be attached to the scenario WFST so that the dialog system behaves naturally as in the corpus. Note that an n-gram model can also be represented as a WFST and composed with other WFSTs.

In the framework of our WFST-based dialog management, several dialog scenarios and functions of system actions are separately designed. This aspect enables us to easily edit dialog structures by adding/removing states and transitions. In addition, we can easily add new input/output symbols in a WFST and prepare functions of system actions for those symbols separately. To accomplish expandability of dialog manager, an individual WFST for each task is prepared and then integrated into a main scenario WFST. Additionally, the dialog scenario for each task consists of task dependent and independent dialogs. Task dependent/independent WFSTs are separately prepared and task independent WFSTs are shared through dialogs.

In the reminder of this paper, we present a general algorithm to manage a dialog system using a WFST, and show a prototype spoken dialog system of Kyoto tour guide with our WFST-based dialog management. This system assists users to make a plan for one-day trip to sightsee through interaction. A WFST for dialog management was constructed based on the

annotated transcript of the Kyoto tour guide dialog corpus we recorded.

2. Weighted Finite-State Transducer based Dialog Management

We use WFSTs for dialog management where input of the WFST is given by a user, which is a word or concept sequence, and then translated into an output sequence using the WFST. Each symbol in the output sequence corresponds to a system action. Although the WFST-based dialog management is basically equivalent to that by the conventional finite-state automaton, it can be designed with more flexibility since there are a lot of useful operations for WFSTs to combine and optimize.

A WFST T over a semiring $\mathbf{K}$ is defined by an 8-tuple as $T = (\Sigma, \Delta, Q, i, F, E, \lambda, \rho)$ where:

(1) Σ is a finite set of input symbols;
(2) Δ is a finite set of output symbols;
(3) Q is a finite set of states;
(4) $i \in Q$ is an initial state;
(5) $F \subset Q$ is a set of final states;
(6) $E \subset Q \times (\Sigma \cup \varepsilon) \times (\Delta \cup \varepsilon) \times \mathbf{K} \times Q$ is a finite set of transitions;
(7) λ is an initial weight;
(8) $\rho : F \rightarrow \mathbf{K}$ is a final weight function.

A meta symbol "ε" indicates there is no symbol to input or output. Figure 1 shows an example of a WFST. The nodes and arcs correspond to states and transitions of the WFST. The label on each arc denotes "input-symbol : output-symbol / weight," and the final state possesses a final weight.

Given an input symbol sequence to a WFST, the output symbol sequence can be obtained as that on the best path with the minimum (or maximum) cumulative weight. The best path can be found efficiently with Dynamic Programming from among successful paths from the initial to one of the finals, which accept the input sequence. In dialog management, however, the system has to respond to the user immediately in each turn. Thus the system needs to choose the most appropriate output sequence according to the current situation. This is the same sense as POMDP. Our WFST-based management includes such a decision process and can also deal with uncertainty during dialog, i.e. when the WFST is non-deterministic, the manager stays at multiple states simultaneously at each turn, which are considered as *hidden* states.

We show the algorithm of our WFST-based dialog management in Fig. 2. Steps 1 to 5 perform initial actions that can be taken by epsilon transitions from the initial state. Steps 6 to 10 respond actions to the user's input at each turn. Steps 11 to 13 check task completion. In the algorithm, π indicates a path consisting of consecutive transitions $e_1, \ldots, e_L$ in the WFST. For a path π, we denote its origin state by $p[\pi]$, its end state by $n[\pi]$, and its cumulative weight by $w[\pi]$ where $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_L]$. "$\oplus$" and "$\otimes$" are two formally-defined binary operations, i.e., "addition" and "multiplication" over the semiring.

In this paper, we use the *tropical semiring* in which the "addition" and "multiplication" of two real-valued weights are defined as the minimum of the two and ordinary addition, respectively. We can also use the *log semiring* in which each weight is defined as a minus log probability, and corresponding "addition" and "multiplication" are defined. Note that a (cumulative) weight is assumed to be better than the others if it is smaller than the others.

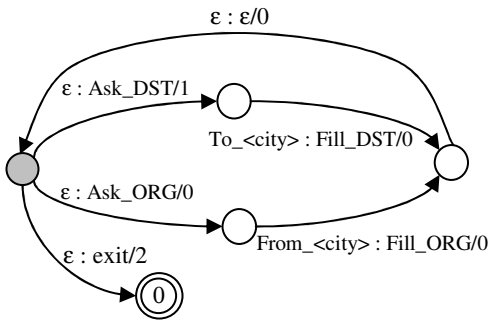

Figure 1: Example of WFST for dialog management. Ask_ORG and Ask_DST indicate system actions to ask the origin and destination cities, respectively. These actions include a meta control that eliminates the transition if the slot has already been filled. Fill_ORG and Fill_DST indicate actions to fill slots according to the user's concept such as From_<city> and To_<city>.

// execute initial actions

1. *// choose the best action sequence in paths with epsilons*
   $\hat{y}_0 \leftarrow \arg\min_{y \in \Delta^*} \bigoplus_{\pi \in P(\{i\}, \varepsilon, y)} \lambda \otimes w[\pi] \otimes C(\pi)$
2. **execute actions corresponding to** $\hat{y}_0$
3. $S_0 \leftarrow \{ s' \mid s' = n[\pi], \pi \in P(\{i\}, \varepsilon, \hat{y}_0) \}$
4. **foreach** $s' \in S_0$ **do**
   $W_0(s') \leftarrow \bigoplus_{\pi : s' = n[\pi], \pi \in P(\{i\}, \varepsilon, \hat{y}_0)} \lambda \otimes w[\pi]$
5. $t \leftarrow 1$

// execute actions for user's input

6. **receive a user's input symbol sequence** x_t
7. *// choose the best action sequence for* x_t
   $\hat{y}_t \leftarrow \arg\min_{y \in \Delta^*} \bigoplus_{\pi \in P(S_{t-1}, x_t, y)} W_{t-1}(p[\pi]) \otimes w[\pi] \otimes C(\pi)$
8. **execute actions corresponding to** $\hat{y}_t$
9. $S_t \leftarrow \{ s' \mid s' = n[\pi], \pi \in P(S_{t-1}, x_t, \hat{y}_t) \}$
10. **foreach** $s' \in S_t$ **do**
    $W_t(s') \leftarrow \bigoplus_{\pi : s' = n[\pi], \pi \in P(S_{t-1}, x_t, \hat{y}_t)} W_{t-1}(p[\pi]) \otimes w[\pi]$

// check task completion

11. $\tilde{W}_F \leftarrow \left\{ \bigoplus_{s' \in S_t \cap F} W_t(s') \otimes \rho(s') \right\} \otimes \left\{ \bigoplus_{s'' \in S_t} W_t(s'') \right\}^{-1}$
12. **if** $\tilde{W}_F < Threshold$ **then exit**
13. *// proceed to the next turn*
    $t \leftarrow t + 1$ **and go to Step 6**

Figure 2: WFST-based dialog management

For a set of states S, input symbol sequence x, and output symbol sequence y, we define $P(S, x, y)$ as a set of all paths each of which originates from one of S, accepts x and outputs y. In Steps 1 and 7, the system selects the most appropriate action sequence on the paths in $P(S, x, y)$. $C(\pi)$ is the expected cost for taking π and the possible future transitions

Table 1: Example of concept and action tags

Concept and action tag	Function	Example utterances or keyword
Grt(start/end)	Greeting for start/end	May I help you? / Have a good trip.
OQ(DST)	Open Question for Destination	Do you have sightseeing plan?
Stt(prf(spot/general))	State preference for spot/general	How about famous temples?
Extrct_kwd	Extract keyword	temple, cherry blossoms, *Kinkakuji*, etc.
Mk_rcmdlist(kwd)	Make a recommendation list	
Rqst(rcmd)	Request recommendation	Do you have any ideas?
Set_rcmdlist	Set a recommendation list	
Set_tgt	Set a target spot	
Rcmd(tgt)	Recommend the target spot	How about *Kinkakuji*?
Agree	State an agree phrase	Wonderful!
Stt(prcs(rcmd))	State next action to recommend	I'm going to list some temples. You can chose some of them.
Expln(tgt)	Explain details of the target spot	*Kinkakuji, Golden Pavilion Temple,* is the informal name of Rokuon-ji.
Cnfrm(dcsn)	Confirm decision	Would you want to go there?
Accept	Accept the recommended spot	Yes, I will go there.
Stt(exprnc)	State experienced or not	I've visited there once.
Stt(imprs(Good/Bad/Next))	State impression such as Good, Bad, Next	Sounds great! / Not so attractive. / Next one please.
Set_imprs(Experienced/Positive/Negative /Neutral)	Set preference parameter	
Rspns2imprs	Response to impression	That's very nice. / Sorry about that.
Prcs4imprs(tgt)	System process based on impression	
Kp(tgt, rcmdlist)	Keep a target in the recommendation list	
Mv(tgt,rcmdlist, dcsnlist)	Move a target to the decision list	
Rmv(tgt, rcmdlist)	Remove a target from the recommendation list	
Stt(next_act)	State next action	I am going to tell you about *Kinkakuji*.
Chck_forloop(rcmdlist)	Check whether for loop is finished or not.	
Trnst(if_forloop_not_end)	Transition	
Trnst(if_forloop_done)	Transition	
Chck_rcmdlist	Check whether recommendation list is empty or not.	
Trnst(if_data_in_rcmdlist)	Transition	
Trnst(if_nodata_in_rcmdlist)	Transition	
Rqst(dcsn4rcmdlist)	Request user' decision	Which spots would you like to go to?
Stt(no_prefered_tgt)	State no preferred spot in the list	Nothing new for me.
Stt(no_requirment)	State no requirement	I have no plans to go there.

from $_n[\pi]$. This is a look-ahead for choosing a more appropriate action at each turn as used in POMDP with (discounted) rewards. S_t is a set of states the system takes at turn t. $W_t(s)$ is the cumulated weight for each state in S_t.

Steps 11 and 12 check the task completion based on $\tilde{W}_F$, the relative overall cumulated weight of all the successful paths. *Threshold* is a pre-defined constant value. In Step 13, the control returns to Step 6 to receive the next user's input.

Generally, a set of (weighted) rules or (hidden) Markov models can be represented as a WFST. Once such a model is embedded into a WFST, it can be combined with other WFSTs. Many useful operations for WFSTs are available to combine and manipulate WFSTs. The composition operation for two WFSTs can be used to generate a WFST that translates sequentially by the two WFSTs.

Suppose we prepare two WFSTs independently, where one translates a word sequence into its corresponding concept sequence for language understanding, and the other translates a concept sequence into system actions for dialog management. If we compose these two WFSTs, the dialog manager can accept word sequences directly using the composed WFST. In addition, some optimization operations are effective to reduce the size and the computational cost in runtime.

Our WFST-based dialog management can be extended to handle uncertainty in speech recognition and/or spoken language understanding. If the user input is represented as a multiple hypotheses such as a lattice, the set of paths $P(S, x, y)$ can also be obtained for the lattice input. The paths may also be weighted with confidence scores. In addition, this extension cannot be used only for such multiple hypotheses, but it can also be used for multi-modal user inputs.

3. Kyoto Tour Guide System

We are developing a Kyoto tour guide system. In this section, we present the Kyoto tour guide dialog corpus we recorded, and a prototype system we have developed so far based on our WFST-based dialog management.

3.1. Kyoto Tour Guide Dialog Corpus

To construct a Kyoto tour guide system, we recorded 50 hours dialog speech and manually transcribed. A 30 minutes dialog was done by a human subject and a professional guide. 100 human subjects played a roll of travelers and planed one-day trip. 2 female and 1 male professionals guided their trip.

In this task, guides assist users to decide sightseeing spots, activities and transportations and then routing and scheduling them. Users do not always have clear ideas for their goals at the very beginning. To clarify users preferences and let users decide their goals through interactions, guides provides sightseeing information proactively, see how much users are interested in the given information and then change strategies to push users to make their decisions based on users' preferences. The strategies to guide sightseeing information are changed by not only users' conditions but also guides' characteristics.

The transcripts of the 5 dialogs were then manually annotated with tags representing user's and guide's actions. The transcripts and their annotation data are used to design a basic dialog scenario and obtain statistics on actions of users and guides, which is used to make the dialog strategy more natural.

3.2. Prototype of Kyoto Tour Guide System

We construct a prototype Kyoto tour guide system using WFSTs. To understand user's intention, a spoken language understanding (SLU) model was constructed as a WFST which translates speech recognition results into concept tags. A scenario WFST was designed manually referring to the real guides' strategies. Figure 3 shows a sample of the scenario WFST which handles tags in table 1. By composing the scenario WFST with the SLU WFST, the resulting WFST can accept a word sequence directly from a speech recognizer, and output an action sequence to be executed by the system.

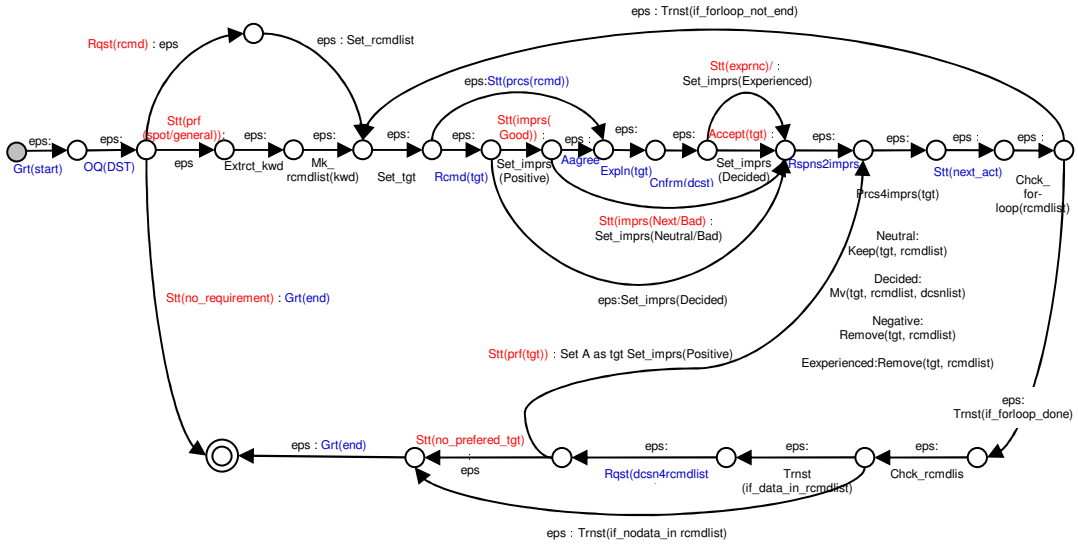

Figure 3: Part of WFST for dialog management in the prototype of Kyoto tour guide system

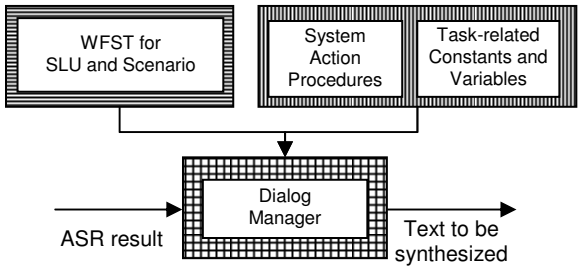

Figure 4: WFST-based dialog system

Table 2: Size of WFSTs

WFSTs	#state	#transition
Scenario	59	146
SLU	152	316
Composed	842	5062
Optimized	546	3311

The WFST was also optimized using weighted determinization and minimization.

Figure 4 shows the configuration of the prototype system. The dialog manager first reads a WFST integrating SLU and dialog scenario. The manager accepts a speech recognition result and translates it into the associated system actions, and then executes the actions and obtains a reply sentence for speech synthesis as a result of the actions. The dialog management algorithm in Fig. 2 was implemented using the MIT WFST toolkit [5].

Table 2 shows the size of WFSTs we prepared for the prototype system, which is measured with the numbers of states and arcs. By composing Scenario and SLU WFSTs, the size of the resulting WFST much increased but it decreased effectively by optimization. We have confirmed our WFST-based dialog management worked well as a Kyoto tour guide while communicating with humans through spoken language interaction.

Since the weights of the scenario WFST has not been trained with our corpus yet, we aim to improve the system so that it behaves more naturally and robustly by using the statistics on concept and action tags annotated in the corpus.

4. Conclusions

This paper proposed an efficient approach to manage a dialog system using a weighted finite-state transducer (WFST) in which concept tags of user's and system's actions are input and output of the transducer, respectively. This framework provides an expandable and adaptable platform of dialog systems. Our pilot task is Kyoto tour guide which assists users to make a plan for one-day trip to sightsee through spontaneous interaction. To construct a Kyoto tour guide system, we recorded 50 hours dialog speech and manually transcribed. We constructed a prototype Kyoto tour guide system using WFSTs and confirmed our approach worked well as a Kyoto tour guide.

Recently, partially observable Markov decision processes (POMDP) has been applied to spoken dialog management [3]. The target of POMDP based dialog systems realizes automatic acquisition of dialog policies through interaction. Such a probabilistic framework for dialog management can also be integrated into our WFST-based dialog management although it could need enormous states. In future work, we'd like to integrate different fashions of scenarios including POMDP.

5. References

[1] J. Glass, "Challenges for spoken dialogue systems," In: Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 1999.

[2] E. Levin, R. Pieraccini and W. Eckert, "Learning dialogue strategies within the markov decision process framework", In Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 72-79, 1997.

[3] J. Williams and S. Young, "Partially Observable Markov Decision Processes for Spoken Dialog Systems." Computer Speech and Language 21(2): pp. 231-422, 2007.

[4] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," Computer Speech and Language, Vol. 16, pp. 69-88, 2002.

[5] L. Hetherington: "The MIT finite-state transducer toolkit for speech and language processing", In *Proc. INTERSPEECH*, pages 2609-2612, 2004.