A DISTRIBUTED ARCHITECTURE FOR COOPERATIVE SPOKEN DIALOGUE AGENTS WITH COHERENT DIALOGUE STATE AND HISTORY

Bor-shen Lin[1], *Hsin-min Wang*[2], *Lin-Shan Lee*[1]

[1]Department of Electrical Engineering, National Taiwan University
[2]Institute of Information Science, Academia Sinica
Taipei, Taiwan, Republic of China
bsl@speech.ee.ntu.edu.tw

ABSTRACT

It has been very difficult to develop spoken dialogue systems with high domain extensibility. Not only the system complexity inevitably increases with the number of topics and domains, but the concurrent topics need to be handled persistently and consistently across different domains. This paper presents a distributed architecture for cooperative spoken dialogue agents with high domain extensibility. Under this architecture, different spoken dialogue agents handling different domains can be developed independently, and cooperate with one another to respond to the user's requests, while a user interface agent can access the correct spoken dialogue agent through a domain switching protocol, and carry over the dialogue state and history so as to keep the knowledge processed persistently and consistently across different domains. Initial experiments provide very encouraging results.

1. INTRODUCTION

In recent years, many spoken dialogue systems have been successfully developed for some specific subject domains [1]. However, not too many of them are able to handle dialogue across multiple subject domains efficiently and intelligently [2]. One of the major difficulties of multi-domain spoken dialogue systems is that the system complexity will naturally be increased with the number of domains handled. When a spoken dialogue system is extended to new domains, significant expansion of the vocabulary size, the grammar rules, the language models, the domain knowledge of the dialogue manager, and the script flow for the sentence generation and so on will all be inevitable. Such expansion automatically degrades not only the processing speed of the system, but also the performance of each module. The word accuracy of the speech recognition module, for example, degrades significantly when the vocabulary size is increased. Another difficulty for such multi-domain systems is that, based on the conventional schemes, the maintainance, modification and extension of such systems are usually not easy. For example, the general-domain language models have to be retrained when we add new vocabularies or new corpora for a new domain. Similar situation occurs to the general-domain parsing rules. Furthermore, with the conventional schemes it's actually impossible to tune one domain without disturbing the others. As a result, a development platform based on a distributed architecture for multi-domain spoken dialogue systems seems attractive. However, for the many topics across different domains with some probable common slots, the most difficult issues include how to maintain the knowledge state of those concurrent topics and how to control the dialogue across different domains. For example, the user may temporarily deviate from one topic, such as hotel reservation, to enter several other topics, such as asking for weather information and train schedule so as to decide the date and time of arrival, and then return to the original topic of hotel reservation and reserve a room. Under a Windows environment with a graphical user interface, this is very natural because a user is free to open multiple sessions with different information servers by simply starting multiple browsers in parallel. However, in a spoken dialogue environment, the user has to deliver all possible information regarding multiple topics and domains sequentially. The system should therefore dispatch the user's requests correctly, and keep the knowledge regarding all different domains coherent, i.e., consistent and persistent.

This paper presents a distributed architecture for cooperative spoken dialogue agents, and the mechanisms for handling the above issues. Under this architecture, all spoken dialogue agents can be developed independently, and then cooperate with one another to achieve the user's goal, while the user interface agent can access these spoken dialogue agents through a domain switching protocol, and carry over the dialogue state and history so as to keep the knowledge persistent and consistent across different domains.

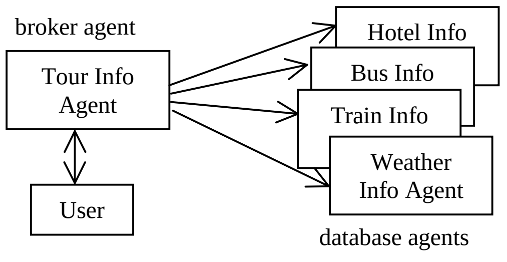


Figure 1. An example of centralized model for tour information service

2. AGENT ARCHITECTURE

The agent technology has been extensively studied and actively developed in recent years. The many practical issues include the agent architectures, and agent languages and so on [3,4]. In this paper, a distributed agent architecture for multi-domain spoken dialogue systems is proposed. Based on this architecture, from the domain knowledge to the dialogue control can all be well distributed over multiple spoken dialogue agents.

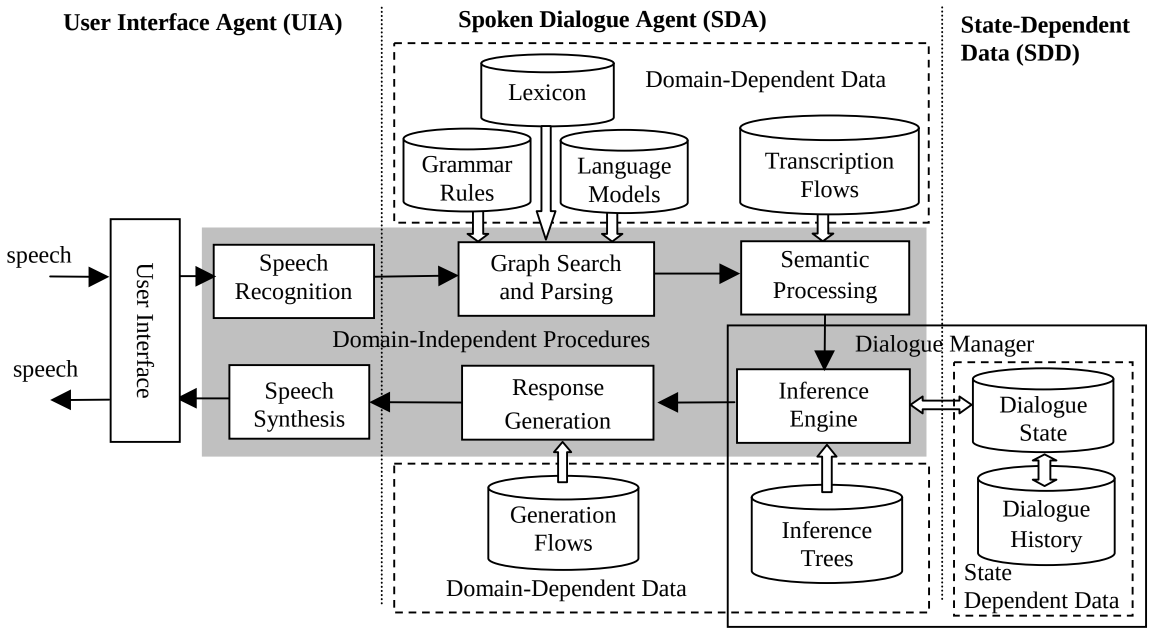


Figure 2. Partition of a spoken dialogue system into a user interface agent, a spoken dialogue agent, and state dependent data

2.1 A Centralized Model

The most straightforward approach for multi-domain information service is a centralized model. As shown in an example in Figure 1, the tour information agent may serve as a broker agent and the hotel, bus, train, and weather information agents as database agents. The broker agent accepts and understands the user's request, sends formatted queries to the database agents, gets the answer, and finally replies to the user. Such a model may be efficient for single-turn queries, but becomes less useful when applied to multi-turn spoken dialogues. In order to handle the dialogue with the user, the broker agent in this model has to be extremely complicated, because the dialogue has to be switched across many different domains smoothly, and therefore the dialogue state and knowledge of all different domains must be kept in the broker agent. Moreover, it will be very difficult to extend the broker agent to new domains.

2.2 A Distributed Model

Because of the potential problems of the centralized model as described above, a distributed architecture based on the client-agent-server model [5] is proposed. Under this architecture, the User Interface Agent (UIA) serves as the client connected to different task agents, or the Spoken Dialogue Agents (SDA), which handle the dialogue and access to the database server so as to respond to user's requests in their respective domains. Figure 2 shows how a typical spoken dialogue system is modularized and partitioned in this model. First, each functional component should be modularized and partitioned into the domain-independent procedure and the domain-dependent data. As shown in Figure 2, the blocks in the shadowed area in the center are the domain-independent procedures, while those enclosed by the dashed lines above or below are the domain-dependent data. For example, the graph search and parsing module is a domain-independent procedure for language understanding, while the lexicon, the language models and the grammar rules are the

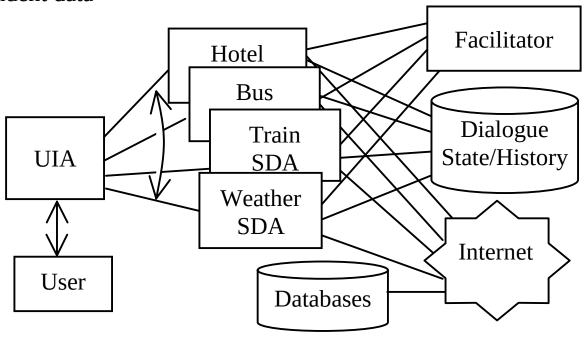


Figure 3. Agent society for spoken dialogue for tour information service

corresponding domain-dependent data. All other functional components are similarly partitioned as well. Besides, the dynamic database of the dialogue manager are also divided into the state-dependent data, including the dialogue state and history [6] also enclosed by the dashed lines on the right of the figure, and the state-independent data, i.e., the inference trees. The state-dependent data expressed in terms of first order logic and partitioned as outside of the spoken dialogue agent (SDA) as shown in Figure 2, thus can be consulted by different spoken dialogue agents for different domains and carried over by the user interface agent when switching across different domains. After all the function components are modularized as shown in Figure 2, a spoken dialogue system is now partitioned into thress parts: a user interface agent (UIA) on the left of the figure, one or more spoken dialogue agent (SDA) each for a specific domain in the middle of the figure, and the state-dependent data (SDD) used across different domains on the right of the figure. The domain-independent acoustic recognition module (including acoustic models and so on) and speech synthesis module (including synthesis units and rules and so on) are in general included in the user interface agent in order to reduce the network bandwidth requirement.

The facilitator decides that $(\mathrm{SDA})_i$ should be switched to $(\mathrm{SDA})_j$

UIA← $(\mathrm{SDA})_i$ switch (new agent $(\mathrm{SDA})_j$, SDD)

  $(\mathrm{SDA})_i$ closes the current dialogue session

UIA→ $(\mathrm{SDA})_j$ initialize-switch

  $(\mathrm{SDA})_j$ initializes a dialogue session

UIA← $(\mathrm{SDA})_j$ ask-state

UIA→ $(\mathrm{SDA})_j$ echo SDD

  $(\mathrm{SDA})_j$ accepts SDD in the dynamic database

UIA← $(\mathrm{SDA})_j$ ask-result

UIA→ $(\mathrm{SDA})_j$ echo the latest recognition result

  $(\mathrm{SDA})_j$ infers next dialogue state

UIA← $(\mathrm{SDA})_j$ response to UIA

Figure 4. Domain switching protocol

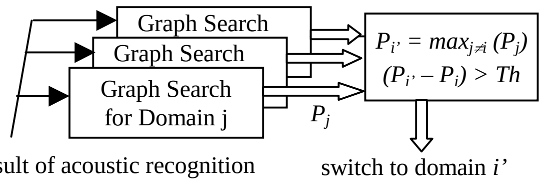
Figure 3 shows an example of the agent society of a spoken dialogue system for tour information services based on the above distributed model. In this model, the domain knowledge and the dialogue control are handled in each SDA instead of the centralized tour information agent as shown in Figure 1. The state-dependent data, the dialogue state and history, are outside of the SDA's so as to be used by all SDA's. The switching across different SDA's, on the other hand, is handled by a facilitator. How these agents cooperate will be discussed in the following sections.

3. COOPERATIVE MULTI-AGENTS

3.1 Domain Switching Protocol

We may use the hospital society to explain the routine of the agent society for a multi-domain spoken dialogue system. Consider the user as a kid, the UIA as the mother, and the many different spoken dialogue agents as the many different doctors in hospital. Then, the dialogue state and history should be with the user no matter which SDA is active, as if the case history should be with the kid no matter which doctor is consulted. When the user's utterance is judged by the facilitator to be of a different domain, the currently active SDA should notify the UIA to switch to that domain, as if a doctor notifies the mother that the kid should be transferred to another doctor. The domain switching protocol is shown in Figure 4 and its associated procedures are presented below.

1. If the facilitator decides that the current active SDA, $(\mathrm{SDA})_i$, should be switched to a new domain handled by $(\mathrm{SDA})_j$, $(\mathrm{SDA})_i$ sends a 'switch' command to the UIA with the attached information, including the address/port of the new agent and the dialogue state/history, and then closes the current dialogue session.
2. The UIA, after receiving the 'switch' command, reconnects to the new agent $(\mathrm{SDA})_j$, and sends the 'initialize-switch' command.
3. The new SDA, $(\mathrm{SDA})_j$, after receiving 'initialize-switch' command, starts a new dialogue session, and then sends a 'ask-state' command to the UIA.
4. When the UIA receives the 'ask-state' command, it sends the dialogue state and history to the new SDA, $(\mathrm{SDA})_j$.
5. When the new SDA, $(\mathrm{SDA})_j$,receives the dialogue state and history, it accepts them to be included in the dynamic database for the dialogue control, and then sends an 'ask-result' command.



Graph Search
Graph Search
Graph Search for Domain j
$P_{i'} = max_{j\neq i} (P_j)$
$(P_{i'} – P_i) > Th$
P_j
result of acoustic recognition
switch to domain i'

Figure 5. Facilitator (current domain *i*)

6. When the UIA receives the 'ask-result' command, it sends the recognition of the user's latest utterance (e.g. phone lattice or similar) to the new SDA, $(\mathrm{SDA})_j$.
7. After the new SDA, $(\mathrm{SDA})_j$, receives the result, it understands it, infers a proper dialogue state, and gives a response to UIA. UIA then transfer the result to the user.

There are several key points should be mentioned. First, the user is not aware of the change of the agents during dialogue, because the UIA hides the details of the negotiation. The knowledge integrity maintained by delivering the dialogue state and history across different domain agents make the user feel that he or she dialogues with a single agent. Second, the dialogue state and history is no longer stored in the SDA after it is disconnected, just like the mother carries over the case history of the kid and registers at a new doctor. Third, in the agent languages like KQML [4], such a mechanism for transferring the dialogue state and history is not well defined yet. For example, in KQML there are performatives for transferring multi-response data, like 'ask-all' or 'stream-all', and for cooperation among agents, like 'broker' and 'recruite'. But it is not necessarily well discussed yet regarding whether or how these performatives can be applied to the cooperation among the spoken dialogue agents such that the 'state of problem-solving' can be kept coherent. Those performatives and protocols seem suitable for single-turn queries rather than multi-turn dialogues.

3.2 Facilitator for Spoken Language

The domain switching is decided by the facilitator. The design of the facilitator for a dialogue system with spoken language is quite different from that for written language, due to the high degree of ambiguities and uncertainties caused by recognition errors. It is not reliable at all for the facilitator to identify the domain by some simple pattern-matching process based on the recognized text strings. Here, a facilitator suitable for dialogues with spoken language is proposed as shown in Figure 5, in which the input is the result of the acoustic recognition, such as the phone lattice or similar, instead of the text string. Graph search is performed for all domains in parallel to generate the optimal path for each domain respectively. Here a path may represent a word sequence, a tag sequence with parsing trees [7], or something similar, based on some search schemes. All the scores of these paths are then sent to a threshold test to decide whether or not to switch the domain. Assume the current domain is domain *i*. The most competing domain, domain *i'*, is then defined as the domain with the highest path score among all the other domains,

$$i' \leftarrow \underset{j\neq i}{arg\ max}(\ P_j\ )$$

where P_j is the path score for the optimal path obtained with domain *j*. The threshold test then decides to switch to the most competing domain *i'* if the difference between the path scores for domain *i* and *i'* exceeds a threshold, whose value is determined empirically as will be discussed in Section 4.

S1: Good morning. How may I help you? (bus agent)
U1: Is there any bus from Taiwan University to the train station?
S2: Here is the bus number from Taiwan University to the train station. (display)
U2: I'd like to reserve the ticket from Taipei to Tainan at 2 p.m tomorrow.
S3: Which train type would you like? (train agent)
U3: A moment. How about the weather of Tainan?
S4: The weather of Tainan tomorrow? (weather agent)
U4: Yes.
S5: The weather of Tainan tomorrow is rainy.
U5: And the day after tomorrow?
S6: The weather of Tainan the day after tomorrow is sunny.
U6: OK. I want two tickets for adult.
S7: Do you want to go the day after tomorrow? (train agent)
U7: Yes.
S8: Which train type would you like?
U8: First class.
S9: Do you want the first class at 2 p.m. the day after tomorrow?
U9: How about the ticket price? . . . . .

Figure 6. An example of multi-domain and multi-topic dialogue

3.3 Concurrent Topics for Multi-Agents

It is important for a multi-domain dialogue system to maintain consistent dialogue across concurrent topics [6]. For a distributed multi-agent architecture proposed here, the control mechanism for topic switching is critical in achieving this purpose. Under such a distributed environment, the SDA which handles each topic must be stored in the dynamic database. In this way, when an active topic is closed, the SDA can immediately identify the new SDA for the newly activated topic. If that domain agent is different from the current SDA, the processes for domain switching as shown in Figure 4 should be executed. Because the dialogue state and history are transferred during the domain switching, this suspended topic can be resumed coherently later on.

4. EXAMPLE SYSTEM AND EVALUATION

We have developed an example dialogue system of distributed agent society for train, weather, and bus information services. All the lexicons, grammar rules, language models, and inference trees for the different SDA's are developed independently, and a common module for domain switching control is augmented to each SDA. Figure 6 is an example of such multi-domain and multi-topic dialogue. As shown in this example, the information for the train ticket reservation was not lost after the user switched to several other domains (from U2 to S9), and the common slots were inherited across domains (from U2 to S4), which show the the persistency of knowledge. The system also detected the inconsistency across domains, invalidated the inconsistent slot (from U2 to S7) and returned to the proper dialogue state (S8), which show the consistency of the knowledge. Apparently, the persistency and consistency of the knowledge are important factors for intelligent multi-domain spoken dialogue systems.

To analyze the performance of domain switching function of the facilitator, 334 utterances of the example system were used for analysis. For each utterance, the path score of the current domain *i*, P_i , and the path score of the most competing domain *i'*, $P_{i'}$, are compared. The false alarm rate can then be derived by counting

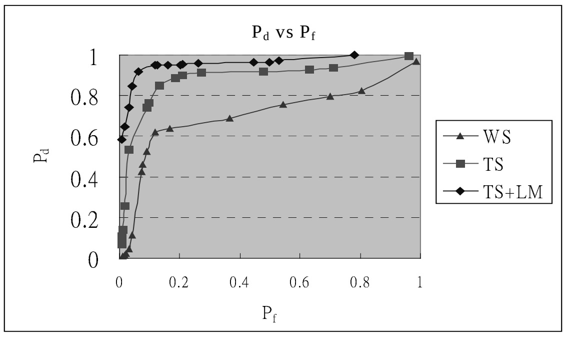


Figure 7. Receiver operator characteristics curves for the detection of domain switching

the utterances in which unnecessary domain-switching occurred, while the detection rate can be derived by counting the utterances in which the desired domain-switching were accurately detected. These detection rate (P_d) and false alarm rate (P_f), which vary with the threshold value described in Section 3.2, give the receiver operator characteristics (ROC) curves for domain switching based on three graph search schemes: word sequence search (WS), tag sequence serach without language model (TS), and tag sequence search with language model (TS+LM), as shown in Figure 7. We can see in this figure how different search schemes with different levels of knowledge affected the detection performance of domain-switching. As can be found in Figure 7, with TS+LM search scheme where grammar rules and language models were integrated, high detection rate can be achieved at very low false alarm rate.

5. SUMMARY

We have proposed a distributed agent architecture for multi-domain spoken dialogue, and developed the cooperative mechanisms for the agent society. This architecture was used to build a multi-domain spoken dialogue system for travenl information services, and very encouraging results were obtained.

6. REFERENCES

[1] Sadek David, Renato De Mori, "Dialogue Systems", in *Spoken Dialogue with Computers*, Chap. 15, 1998.

[2] Seneff S., etc., "GALAXY-II: A Reference Architecture for Conversational System Development", SL981153.PDF, *ICSLP98*.

[3] Wooldridge M., Jennings N., "Intelligent Agents: Theory and Practice", in *Knowledge Engineering Review*, Vol. 10, No. 2, Cambridge University Press, 1995.

[4] Finin Tim, Labrou Yannis, and Mayfield James, "KQML as an agent communication lagnuage", in *Software Agents*, MIT Press, 1997.

[5] Jane Yung-jen Hsu, "A Multi-Agent Framework for Intranet Service Integration", *Lecture Notes in Artificial Intelligence, Vol.1599, PRIMA'98: Multiagent Platforms*, T. Ishida (Ed.), Springer-Verlag, 1999, pages 148-161.

[6] Lin, B. S., Wang, H. M., and Lee, L. S., "Consistent Dialogue Across Concurrent Topics Based on an Expert System Model", *to appear in EUROSPEECH99*.

[7] Lin B. S., Chen Berlin, Wang H. M., and Lee L. S., "Hierarchical Tag Graph Search for Spontaneous Speech Understanding in Spoken Dialogue Systems", SL980449.PDF, *ICSLP98*.