Dynamic Help Generation by Estimating User's Mental Model in Spoken Dialogue Systems

Yuichiro Fukubayashi, Kazunori Komatani, Tetsuya Ogata, Hiroshi G. Okuno

Graduate School of Informatics, Kyoto University
Yoshida-Hommachi, Sakyo, Kyoto 606-8501, Japan.
{fukubaya,komatani,ogata,okuno}@kuis.kyoto-u.ac.jp

Abstract

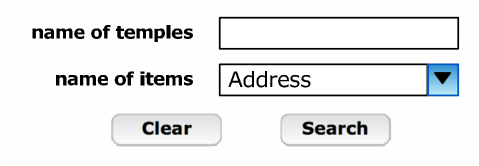
In a speech interface, a gap between a user's mental model and actual structures of systems tends to be large because the amount of information conveyed by speech is limited. We address dynamic help generation adapted to users, to decrease the gap between them. We defined a domain concept tree as an expression of a system's actual structure. We estimated and maintained user's knowledge about the system on the tree. Every node in the tree has values representing the degree to which a user understands the concepts corresponding to the nodes. The values are updated based on the content of user's utterances and help messages the system gives. Help messages provided for users are determined by referring to the domain concept tree and identifying concepts the user does not understand. We evaluated our method by testing twelve novice subjects. Both the average time to complete tasks and the number of utterances significantly decreased because of the help messages provided by our method.

Index Terms: spoken dialogue system, adaptive help generation, novice user

1. Introduction

Spoken dialogue systems are especially helpful in situations without a display, such as a telephone or an interface with robots. In such an interface, a lot of information cannot be conveyed quickly to users because the only medium such systems can use is speech. For example, when instructing a user what type of input the system expects, an interface with a display such as a web browser can intuitively convey a lot of information by showing examples or input forms visually. On the other hand, in a speech interface, the system needs to verbally speak the explanation and examples one by one, which takes a long time, and also the user often cannot remember all the content. That is, since speech media are sequential, contents conveyed to a user must be selected. Moreover, since speech media are evanescent, that is, it disappears immediately, it is important to select situations in which systems output information to keep it in the user's short-term memory. Considering these characteristics, an appropriate design is needed so that novices can use a speech interface without difficulty.

To give users a correct mental model, the system should generate appropriate help messages that decrease the gap between a user's mental model of the system and the system itself. Especially, novice users need help messages in a spoken dialogue system [1]. To generate such a help message, we need to express the actual structure of a system and the user's mental model of it, which varies during dialogues. Although some studies focused on generating help messages [2, 3], they did not take into consideration changes in the user's knowledge during the dialogue, and generated same help messages if the same speech recognition results were obtained.



name of temples:
name of items: Address

Figure 1: Example of web search interface

In this study, we defined a domain concept tree that represents the actual structure of a system. We represented a user's mental model of the system by giving values to each node on the tree, which corresponds to each concept of the system. We estimated and maintained the user's mental model of the system on the tree, enabling us to generate appropriate help messages.

2. Mental Model and Help Messages in Spoken Dialogue Systems

2.1. Mental Model and Role of Help Messages

A mental model represents user's knowledge about a system. To make a system easy to use, we should decrease the gap between the user's mental model and the actual structure of the system. Generally, this gap tends to be larger in spoken dialogue systems because, especially in systems that handle only speech, the systems initially provide little affordance [4]. For example, in a web search interface, like that shown in Figure 1, users can easily guess what kind of information is required for each slot. In a spoken dialogue system, however, it is hard for the system to provide the same affordance, and novices are at a loss as to what to say.

In spoken dialogue systems, appropriate generation of help messages are indispensable to give appropriate affordance for the systems and to decrease the gap between a user's mental model and the actual structure of the system. To generate such a help message, the system must strictly select the content of the help message and when to provide it according to user's knowledge because the amount of information included in speech is limited.

U1:	Tell me about Kiyomizu temple. *(FILLER FILLER Kiyomizu temple)*
S1:	I could not understand what you want to know about “Kiyomizu temple”. To get information about temples, ask me a question with both a name of temples
U2:	Tell me the history of Kiyomizu temple. *(FILLER FILLER Kiyomizu temple)*
S2:	I could not understand what you want to know about “Kiyomizu temple”. You can get information about temples, for example, by saying “summary”, “address”, “telephone number”, as a name of items.

An *italic* sentence in ( ) denotes speech recognition results.

Figure 2: Example dialogue

2.2. Strategy for Help Generation Reflecting User’s Knowledge

Help messages should be hierarchically composed as to the degree of a user’s understanding because the degree is different among users. For novices who know nothing about the system, the system should provide help messages at an abstract level. Specifically, help messages should provide an outline of the system, for example, “This is the Kyoto sightseeing guide system. You can search for and get information about temples in Kyoto”. Then, it should provide help messages that teach more concrete question patterns gradually, such as “for example, if you say ‘tell me the address of Kiyomizu temple’, you can get the information”. It is expected that this strategy will increase efficiency throughout the dialogue because it avoids forthcoming out-of-domain utterances. On the other hand, providing help messages that are already known by the user is redundant. For example, if a user knows concrete question patterns, help messages teaching those are useless.

Figure 2 shows an example of a dialogue that takes into account the two points of the strategy. In utterances **U1** and **U2**, the system could get only the content word “Kiyomizu temple” from the speech recognition result. However, since the system explained the outline in **S1**, the system considered that the user knew the outline. So the system provided the help message teaching concrete phrases in the system in **S2**.

2.3. Assumption about User’s Knowledge

We make an assumption to generate dynamic help messages reflecting user’s knowledge. We assume that a user who understands a certain concept also understands abstract concepts of it. Conversely, we assume that a user who does not understand a certain concept also does not understand more concrete concepts of it.

For example, we assume that users who understand concrete question patterns such as “tell me the address of Kiyomizu temple” also understand its abstract concepts like that the system can provide information about temples. We also assume that users asking out-of-domain questions may know neither the domain the system treats nor a more concrete concept like that the system provides information about temples.

3. Dynamic Help Generation Based on a Domain Concept Tree

We address generation of help messages that can decrease the gap between a user’s mental model of a system and the actual structure of it. The messages should change in the dialogue, adapting to the user’s knowledge. To generate such messages, we need to resolve the following issues:

1. how to represent the concept structure of an actual system,
2. how to estimate a user’s knowledge and how to maintain that knowledge varying through the dialogue, and
3. how to determine the content of help messages to reflect the user’s knowledge.

3.1. Domain Concept Tree

To represent the concept structure of a system, we designed a tree representing the hierarchal layers of concepts in the domain. The domain concept tree for the Kyoto sightseeing guide system is shown in Figure 3. Here, node names in the content word layer represent concrete phrases contained in the system vocabulary. The domain concept tree is composed of four layers; in the tree, the more abstract concepts are placed on the upper layers, and the less abstract ones on the lower layers. The concept representing the whole system is placed on the top (root node), and the concepts representing words contained in the system vocabulary are on the bottom (leaf nodes). The four layers are

the system layer representing the whole system,
the function layer representing each function provided by the system,
the element layer representing each element that should be included in utterances to use the functions,
the content word layer representing individual words and phrases making up each element.

We define two kinds of nodes, AND and OR, to reflect the grammatical constraints in spoken dialogue systems. The AND nodes are in the function layer, which require that all children nodes in the element layer are completed. This means that the system has a grammar rule in which all these elements should be contained in a single utterance. If an AND node has only one child node, it is equal to an OR node. On the other hand, OR nodes are in all layers. The following example illustrates the characteristics of AND and OR nodes.

In the Kyoto sightseeing guide system, whose domain concept tree is shown in Figure 3, the element nodes of the children of the “get information about temples” node have AND relations with each other. For example, when a user wants to get information about temples, a user’s utterance should contain both <names of temples> and <names of items>. If one of the elements is missing, the user cannot get the information in this system. Accordingly, we cannot say that a user knows how to use the function until the user knows that two words are needed when getting information about temples. In this meaning, “get information about temples” is an AND node, and its children “names of temples” and “names of items” in the element layer, are in the AND relation.

In this system, the functions “repeat”, “get information about temples”, and “search temples by conditions” are available anytime. When the user wants to get information about temples, he/she selects one word for <names of items> from “address”, “telephone number”, and so on, and asks the system a question. In this meaning, the children of “Whole system” or “names of items” are in the OR relation, respectively.

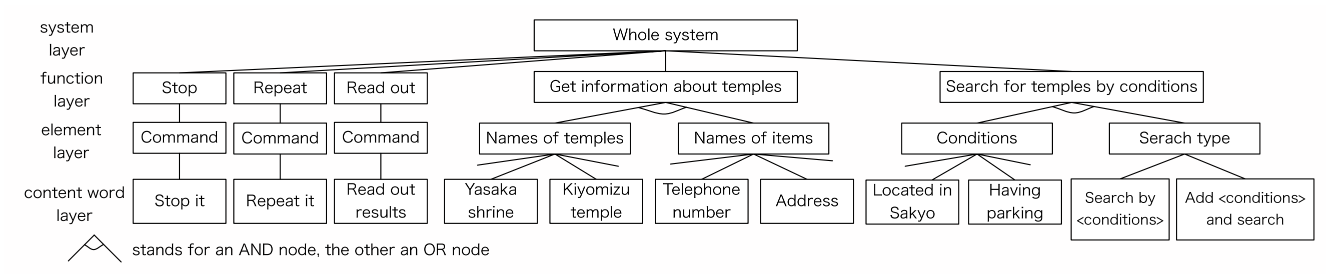


Figure 3: Domain concept tree of the Kyoto sightseeing guide system

3.2. Estimation of User's Mental Model and its Maintenance on Domain Concept Tree

We represent user's mental model on the domain concept tree. Each node in a domain concept tree has values (known degrees), which represent the degree of how much a user understands concepts corresponding to the nodes. A known degree ranges from 0 to 1. A known degree over 0.5 means that a user understands the concept, and one under 0.5 means he/she does not understand. That is, smaller known degree denotes larger necessity of help for users. The initial value of a known degree is 0.5, which means neutral. We estimate a user's knowledge by updating the known degree of each node based on the content of user's utterances and help messages the system gives.

We defined $\texttt{renew}(\boldsymbol{n}, p)$ to update the known degrees. Function `renew` increases the known degree of node $\boldsymbol{n}$ and its ancestors if $p > 0$, and decreases that of $\boldsymbol{n}$ and its descendants if $p < 0$. `renew` uses parameters $p_u^+ (> 0)$, $p_h^+ (> 0)$, and $p_u^- (< 0)$ for updating the known degrees ($-0.5 \le p_{h,u}^{+,-} \le 0.5$). The subscript u means updating based on user's utterances, and subscript h means updating based on help messages from the system. Furthermore, we incorporate a factor for updating degree p. The factor is denoted by λ ($0 \le \lambda \le 1$). The degree of updating gets smaller (p, λp, $\lambda^2 p$, $\cdots$) as the upper (or lower) nodes are updated. The degree of updating gets smaller by λ because of an indeterminateness in the assumption in section 2.3. In other words, if a user understands a concept, he/she may understand (does not understand) the upper (lower) concepts, but this assumption is not certain.

Function $\texttt{renew}(\boldsymbol{n},\, p_u^+)$ and $\texttt{renew}(\boldsymbol{n},\, p_u^-)$ are executed in following two steps:

1. updating according to content words in a speech recognition result, and
2. updating by checking if an AND condition is satisfied.

First, when there are some content words in a speech recognition result, $\texttt{renew}(\boldsymbol{n}, p_u^+)$ is executed for the corresponding nodes $\boldsymbol{n}$ in the content word layer, and the known degrees of the nodes of their ancestors increase. When there are no content words, $\texttt{renew}(\boldsymbol{n},\, p_u^-)$ is executed after setting the root node to $\boldsymbol{n}$, and the known degrees of the nodes of their descendants are decrease.

Then, AND nodes that are ancestors of the content word nodes contained in the speech recognition result are checked to determine whether their AND conditions are satisfied. The system assumes that users understand the concept of the AND node only if the AND condition is satisfied. Therefore, after setting the AND node to $\boldsymbol{n}$ that is an ancestor of nodes corresponding to the words in the speech recognition result, $\texttt{renew}(\boldsymbol{n},\, p_u^+)$ is executed only when all the elements of $\boldsymbol{n}$ are contained in the speech recognition result. If there are missing elements, $\texttt{renew}(\boldsymbol{n},\, p_u^-)$ is executed for the AND node $\boldsymbol{n}$.

Function $\texttt{renew}(\boldsymbol{n},\, p_h^+)$ is executed for node $\boldsymbol{n}$ whose concept is provided as a help message. When function `renew` is executed for a node in the function layer, $\texttt{renew}(\boldsymbol{n},\, p_h^+)$ is also executed for nodes in the content word layer corresponding to words used as examples in the help message provided. When a help message contains N content words, $\texttt{renew}(\boldsymbol{n}, p_h^+/N)$ is executed for each node $\boldsymbol{n}$ corresponding to the content words. The degree of updating of known degree p is divided by N because the probability that a user remembers N content words is less than that the user is provided only one content word.

3.3. Help Generation based on Domain Concept Tree

Our system provides a help message in two cases:

- when a user's utterance is incomplete and cannot trigger a function of the system (missing elements of an AND node), and
- when a user remains silent.

In the former case, the content of the help message is determined after selecting a starting node and searching for the most appropriate message from the descendants of the starting node. The starting node is the one for which $\texttt{renew}(\boldsymbol{n},\, p_u^-)$ was executed when updating known degrees. The candidate nodes for the help message meet the following conditions:

1. They are descendants of the starting node (including the starting node itself).
2. The known degrees of the node are under 0.5.
3. They are in the upper most layer that satisfies requirements 1 and 2.

If there are many candidate nodes, the system selects a node whose known degree is the minimum. If there are many nodes whose known degrees are equally the minimum, the system selects one of the candidate nodes at random. Then, help messages are generated by using templates of the selected node, shown in Figure 4 as examples.

In the latter case, a help message is generated when the silence continues for t seconds. The content of the message is determined by the same procedure as the former case, after assuming the root node as the starting node.

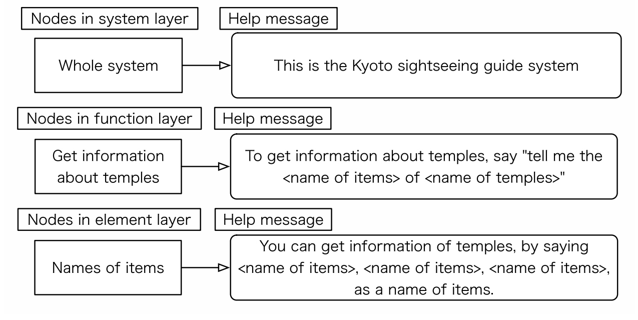

Nodes in system layer
Help message
Whole system
This is the Kyoto sightseeing guide system
Nodes in function layer
Help message
Get information about temples
To get information about temples, say "tell me the <name of items> of <name of temples>"
Nodes in element layer
Help message
Names of items
You can get information of temples, by saying <name of items>, <name of items>, <name of items>, as a name of items.

Figure 4: Example of help messages for each node

4. Implementation and Evaluation

4.1. Implementation

We implemented our method into the Kyoto sightseeing guide system. Its dialogue management is executed in user initiative, and the system replies to the user's question only in speech. The system with our method outputs a help message if the question is incomplete and the system cannot make a query, after an error message reflecting the speech recognition result (e.g., I could not understand what you want to know about "Kiyomizu temple"). A help message is not effective and knowledge model differs from actual if a user does not listen to the end. Therefore, when the system is speaking, user's utterance is ignored.

We used Julian[1] as the speech recognizer. The vocabulary size is 673. The database of the system is composed of 279 entries, and each entry has 16 keys, such as address, telephone number, and so on. We set continuous silent time t as 15 seconds. The update parameters of the domain concept tree are set as $p_u^+ = 0.25$, $p_h^+ = 0.2$, $p_u^- = -0.05$, and $\lambda = 0.5$.

4.2. Experimental Evaluation

We evaluated our method using novices who had never used spoken dialogue systems and were given no instruction about the system's usage. Twelve subjects were given two tasks, A and B, and were first asked to complete task A, and then task B. These two tasks are similar, except for content words used in them. Half of the subjects (Group 1) used a system that provided helps based on our method. After that, they used the system that provided no help. The other half of the subjects (Group 2) used the systems in reverse order. Tasks were regarded as incomplete if a user could not accomplish them within 25 minutes. When giving scenarios, we did not use the same words or phrases as in the system grammar, to prevent them from becoming instruction.

Table 1 shows the task success rate, the average duration, and the number of utterances to complete tasks. The data when tasks were not completed are not included in average duration and the number of utterances. In both groups, all subjects accomplished tasks when using the system providing help messages, but a few subjects could not when using the one with no help message. In task A, namely, in the former half of the experiment, subjects of Group 1 who were provided help messages performed better than those of Group 2 provided no help message, in task success rate, duration, and the number of utterances. This result shows that generated help messages were helpful to complete the task, especially when subjects were not accustomed to the system. In the latter half of the experiment, there was no difference between the two groups, because subjects were accustomed to the system enough.

Table 1: experimental result (data when tasks were not completed is not included in average duration and # of utterance)

	task (system)	succ. rate	duration (sec.)	#utt.	ASR corr.
Group 1	A (H)	6/6	926	50.5	51.4%
(H → NH)	B (NH)	5/6	575	39.2	50.5%
Group 2	A (NH)	3/6	1191	117.0	27.0%
(NH → H)	B (H)	6/6	688	39.5	42.8%

H: system provided help messages
NH: system provided no help messages

Average word correctness of ASR was 42.9%. The correctness in task A of Group 2 was 27.0%, which was especially low. This was because there were many utterances that cannot be accepted by the system because subjects had given no instruction about the system's usage. Some subjects could not complete their tasks when they could not find appropriate question patterns that can be accepted by the system. In such cases, it is helpful for the system to show users some expressions that can be accepted.

5. Conclusions

We defined a domain concept tree as an representation of a system's structure, to decrease the gap between a user's mental model and the system itself. We estimate and maintain the user's knowledge about the system on the tree. Helps provided for users are determined by referring the tree and searching a concept estimated as the user does not know. We implemented our method into the Kyoto sightseeing guide system and showed its effectiveness, especially for novice users. There are still some issues as our future work: setting optimal parameters described in Section 4.1, extending our framework to mixed-initiated dialogue management, and adapting not only content of help messages but also surface expressions of them to the known degree. Now we are also trying to introduce utterance verification technology into our framework, for coping with situations where there are many out-of-domain utterances.

6. References

[1] D. Bohus and A. I. Rudnicky, "Sorry, I didn't catch that! - An investigation of non-understanding errors and recovery strategies," in *SIGDial*, 2005.

[2] G. Gorrell, I. Lewin, and M. Rayner, "Adding intelligent help to mixed-initiative spoken dialogue systems," in *Proc. ICSLP*, 2002.

[3] B. A. Hockey, O. Lemon, E. Campana, L. Hiatt, G. Aist, J. Hieronymus, A. Gruenstein, and J. Dowding, "Targeted help for spoken dialogue systems," in *Proc. EACL*, 2003.

[4] D. A. Norman, *The Psychology of Everyday Things*, Basic Books, 1988.

[1]http://julius.sourceforge.jp/