PAPER

# Selecting Help Messages by Using Robust Grammar Verification for Handling Out-of-Grammar Utterances in Spoken Dialogue Systems**

**Kazunori KOMATANI**[†*a)], ***Member***, **Yuichiro FUKUBAYASHI**[†], **Satoshi IKEDA**[†], **Tetsuya OGATA**[†], ***and*** **Hiroshi G. OKUNO**[†], ***Nonmembers***

**SUMMARY** We address the issue of out-of-grammar (OOG) utterances in spoken dialogue systems by generating help messages. Help message generation for OOG utterances is a challenge because language understanding based on automatic speech recognition (ASR) of OOG utterances is usually erroneous; important words are often misrecognized or missing from such utterances. Our *grammar verification* method uses a weighted finite-state transducer, to accurately identify the grammar rule that the user intended to use for the utterance, even if important words are missing from the ASR results. We then use a ranking algorithm, RankBoost, to rank help message candidates in order of likely usefulness. Its features include the grammar verification results and the utterance history representing the user's experience.

***key words:*** *spoken dialogue system, help generation, out-of-grammar utterances, novice user, utterance history*

## 1. Introduction

Spoken dialogue systems are being moved from the laboratory to the public sphere [2]–[4], so opportunities are increasing for the general public to use such systems. Even novice users can now directly access such systems without previous instruction, which is quite different from laboratory experiments in which instructions are normally given to users. Novice users often input utterances that are not correctly recognized due to the gap between the system and the user's mental model of it, i.e., the user's expectations about the system. Thus, user utterances often cannot be interpreted correctly because of the system's limited grammar for language understanding (LU). We call such utterances "out-of-grammar (OOG) utterances". The user needs to change such utterances to make them acceptable, but cannot do so unless they are given instructions on what expression patterns are accepted by the system.

Our approach to managing the problem of OOG utterances is to provide the user a help message showing an example of an acceptable language expression when the user's utterance is not acceptable. For example, in a sightseeing task, if a user wants to get information on a tourist spot but an automatic speech recognition (ASR) error occurs because of an OOG utterance, a help message would be generated adaptively; for example, "If you want to get information on a tourist spot, you can say, 'Tell me the phone number of Kiyomizu Temple', for example, and specify the name of the place". We prepare such help messages that corresponds to each grammar rule the system has. We therefore assume that an appropriate help message can be provided if the user's intention, i.e., the grammar rule that the user intended to use for his utterance, is correctly identified.

Requirements for generating such help messages would include:

1. identifying the user's intention even from OOG utterances and
2. considering utterance history that represents the user's experience.

The first requirement is based on the fact that ASR results, which are the main input to the spoken dialogue system, are usually erroneous for OOG utterances. Identifying the grammar rule that the user intended to use is accordingly difficult especially when content words, which correspond to database entries such as place names and their attributes, are not correctly recognized. To identify the user's intention from erroneous ASR results, all three types of ASR errors (insertion, deletion, and substitution) in any position should be taken into consideration. The second requirement is based on the fact that an ASR result for an OOG utterance does not necessarily contain sufficient information to identify the user's intention. This is because of ASR errors or because users tend to omit some elements from their utterances when they are clear in the context. Therefore, the information needs to be complemented by using the user's utterance history to provide appropriate help messages.

We develop a grammar verification method based on a weighted finite-state transducer (WFST) to meet the first requirement. It robustly identifies the grammar rule intended for the utterance even for OOG utterances. A WFST representing an ASR result takes all possible errors into consideration. To meet the second requirement, we incorporate the utterance history representing the user's experience and the user's knowledge [5] as features of a boosting algorithm, RankBoost [6]. It ranks help message candidates in order of

Manuscript received April 5, 2010.
Manuscript revised July 6, 2010.

[†]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
[*]Presently, the author is with the Graduate School of Engineering, Nagoya University, Nagoya-shi, 464–8603 Japan.

[**]This paper is a modified and extended version of an earlier report [1].

a) E-mail: komatani@nuee.nagoya-u.ac.jp

DOI: 10.1587/transinf.E93.D.3359

likelihood. Because it is difficult even for human annotators to uniquely determine which help message should be provided for each case, we use a ranking algorithm that ranks help message candidates on the basis of training examples with multiple labels having a certain order of priority.

## 2. Related Work

A robust system should be able to handle situations in which a user's input cannot be correctly interpreted by the system. Dzikovska et al. [7] analyzed and categorized interpretation errors in their text-based tutoring system. Bohus and Rudnicky [8] analyzed non-understanding errors in spoken dialogue systems. One way to prevent such errors is to acquaint users with the limitations of the system through a preceding tutorial session [9]. Tomko et al. [10] designed a standardized protocol called "Speech Graffiti". They designed a subset language used in spoken dialogue systems and taught the user how to use it in a short tutorial session. They also tried to "shape" user utterances by changing the expressions used by the system for confirmation [11]. We take the approach of guiding the user's utterances to match the system's capability without providing a tutorial session before using the system. We did this by designing the system to provide adaptive and explicit help messages during dialogues.

There have been a number of studies on generating help messages in spoken dialogue systems. Gorrell et al. [12] trained a decision tree to classify causes of errors for OOG utterances. Hockey et al. [13] classified OOG utterances into three categories (endpointing errors, unknown vocabulary, and subcategorization mistakes) by comparing two ASR results. Their method, called "Targeted Help", provides a user with immediate feedback tailored to what s/he said. Lee et al. [14] addressed error recovery by developing a method to generate help messages in an example-based dialogue modeling framework. These approaches, however, determine the help message to provide mainly on the basis of literal ASR results and do not take the user's utterance history into consideration. Therefore, the messages are degraded if the ASR results have a lot of information missing, especially for OOG utterances.

An example dialogue enabled by our method, especially the part of the method described in Sect. 4, is shown in Fig. 1. The user utterances are transcriptions, and the "S" and "U" denote system and user utterances, respectively. In this example, the ASR results for the user utterances (U1 and U2) do not contain sufficient information because the utterances are short and contain out-of-vocabulary words. These two results are similar, so the help message after U2 provided by methods like Targeted Help [12], [13] will be the same as Utterance S1 because they are based only on the ASR results. Our method can provide different help messages (e.g., Utterance S2) after ranking candidates by considering the user's utterance history and grammar verification results. Because the candidates are arranged in the order of likely usefulness, the most appropriate help message can be provided after fewer user attempts.

**U1: Tell me your recommended sites.**
*Underlined parts are not in-vocabulary, and no valid LU result is obtained. The identified grammar rule is [Obtaining info on a site] although the most appropriate help message is that corresponding to [Searching tourist sites].*

**S1: I did not understand. You can say, "Tell me the address of Kiyomizu Temple", for example, if you want to get information on a site.**
*The help message corresponding to [Obtaining info on a site] is provided.*

**U2: Tell me your recommended sites.**
*The user repeats the utterance probably because the help message (S1) was not very helpful. The identified grammar rule is [Obtaining info on a site] again.*

**S2: I did not understand. You can say, "Search shrines or museums", for example, if you want to search for tourist sites.**
*Another help message corresponding to [Searching tourist sites] is provided after ranking candidates by using the user's utterance history and the grammar verification result.*

[ ] denotes grammar rules.

**Fig. 1** Example dialogue enabled by our method.

This method for ranking help message candidates is also useful in a multimodal interface with speech input. Help messages are necessary when ASR is used as its input modality; for example, such messages are implemented in City Browser [15]. This system lists template-based help messages on the screen by using ASR results and the internal states of the system. The order of help messages is important, especially in portable devices with a small screen, on which the number of help messages displayed at one time is limited, as Hartmann and Schreiber pointed out [16]. Even if large screens are available, too many help messages without any order will distract the user's attention and thus spoil the system's usability.

We assume that an appropriate help message can be provided by identifying the grammar rule that the user intended to use, which corresponds to the user's intention. In other words, help message generation is one application realized by identifying the user's intention. Because our goal is to generate help messages for language expressions the user does not know, we define user intention not by slot values but by grammar rules corresponding to the expressions. There have been several studies on identifying the user's intentions as slot values by using history information and treating multiple candidates derived from ambiguous ASR results. In such studies, the most appropriate slot value was selected after maintaining multiple candidates using rules for tree structures that maintain histories [17], a particle filter [18], or statistics on dialogue acts and dialogue state up-

dates [19]. History information has also been used to estimate confidence measures for slot values in a current user utterance [20], [21]. In these studies, ASR results for OOG utterances were simply discarded if they did not contain any slot values; that is, no intention was identified.

In contrast, the grammar verification method we develop can identify the user's intention even from ASR results containing no correct slot value. This is possible because we define user intentions as grammar rules, which are less concrete than slot values but are helpful in generating help messages. Furthermore, the features used to identify user intention are different from those in the related studies mentioned above. We use features representing the user's experience such as whether or not the user has already uttered the language expression, that is, whether the user already knows the expression. They are used to identify user intention at the grammar level for generating help messages, while related studies focus on the reliability of slot values.

## 3. Grammar Verification Based on WFST

As mentioned, we identify the user's intention as the grammar rule that the user intended to use for his/her utterance even if it is OOG. We call this method *grammar verification*. We use two kinds of transducers: finite-state transducers (FSTs), representing each task grammar, and weighted FSTs (WFSTs), representing each ASR result and its confidence score. Hereafter, we denote them as "grammar FST" and "input WFST". The grammar verification (GV) score is defined as the sum of the weights of the sequence obtained by composing the input WFST and the grammar FSTs. The optimal grammar rule is the sequence having the maximum GV score.

Our method takes all three types of ASR errors into consideration. The input WFST is designed to be able to generate arbitrary sequences in which every word in an ASR result can be replaced by an insertion or substitution error and in which any word can be deleted. Its weight is designed to reflect the confidence score of the ASR result and the word length. Accordingly, we obtain the grammar rule that is nearest to the input ASR result even if an element in it is misrecognized or absent. An overview of the grammar verification method is depicted in Fig. 2.

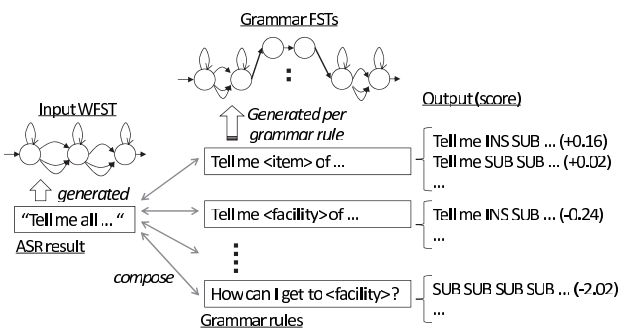



**Fig. 2** Overview of grammar verification method.

### 3.1 Design of Input WFST and Grammar FST

In the input WFST and grammar FST shown in Fig. 3, each state transition has a label in the form of "a:b/c" denoting input symbol a, output symbol b, and weight c. Weights are omitted in the grammar FST because no weights are used[†]. Input symbol $\varepsilon$ indicates a state transition without any input symbol, that is, an epsilon transition. Output symbol $\varepsilon$ indicates no output in the state transition. For example, a state transition "please:$\varepsilon$/1.0" is executed when the input symbol is "please"; in this case, no output symbol is generated, and 1.0 is added to the GV score.

The input WFST is automatically constructed from the ASR result. Words in the ASR result make up sequential state transitions, and each of them is paralleled by filler transitions: INS and SUB. Another filler transition, DEL, is attached to every state as a self loop. The filler transitions INS, DEL, and SUB are assigned to each state to represent the errors, insertion, deletion, or substitution. All input symbols in the input WFST are $\varepsilon$, with which the WFST represents all possible sequences containing arbitrary errors. For example, the input WFST in Fig. 3, in which the ASR result is "Every Monday please", represents all possible sequences including fillers such as "Every Monday please", "Every Monday *F*", and "*F* Monday *F*", where every word can be replaced by symbol *F* that represents an insertion (INS) or substitution error (SUB). Moreover, the error symbol DEL can be inserted into the output symbol sequence at any position; it corresponds to a deletion error in the ASR result. The weights for each state transition are summed up to determine the GV score, and the optimal result is determined. The weights are explained in Sect. 3.2.

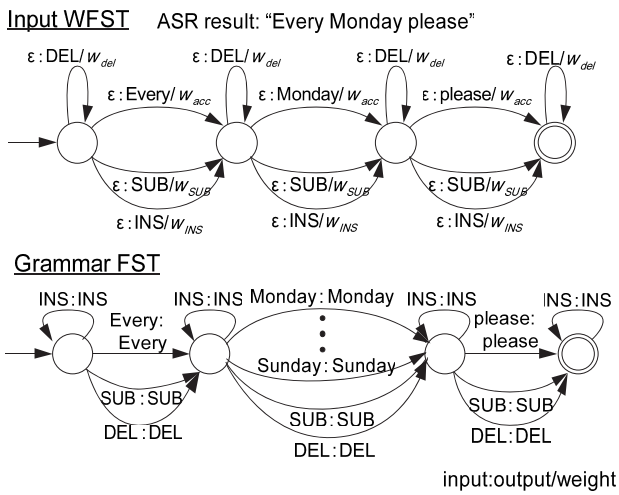



**Fig. 3** Example of input WFST and grammar FST.

[†]The grammar FST actually has a few weights in the current implementation to calculate weights at runtime. Specifically, only arcs of SUB and DEL have weights corresponding to the word length of its grammar element used in Eqs. (3) and (5). We denote it as "FST" for conceptual simplicity.

The grammar FSTs are generated from the task grammar, which is written by a system developer for the target domain. They determine whether an input sequence conforms to the task grammar. Filler transitions are assigned to each state to handle each type of ASR error that might be in the input WFST. A filler transition, INS, DEL, or SUB, is added to each state in the FST, except for states within keyphrases as indicated by the system developer. In the example shown in Fig. 3, the grammar FST accepts "SUB Monday please" but does not accept "DEL SUB Monday please", both of which are outputs of the input WFST.

### 3.2 Weights Assigned to WFST

The GV score represents how close an ASR result is to each grammar rule in the system. The score is defined as the sum of the weights for all the $i$-th output symbols of the composed WFST and is calculated for each grammar rule $k$:

$$\text{GV score}_k = \max_j \sum_i w_{ijk}, \tag{1}$$

where $j$ denotes the output sequences of the composed WFST. The $i$-th output symbol is obtained as part of output sequence $j$ after taking account of possible correspondences between grammar rule $k$ and the current ASR result. The sequence with the highest score is obtained after decoding by the WFST in which possible output sequences $j$ are considered.

Weight $w_{ijk}$ is given to each symbol of WFST output $i$ so that its value becomes larger when the current ASR result is closer to grammar rule $k$. The weight is defined by using the products of the confidence score and word length of the current ASR result [22]. This means that an ASR result is regarded to be closer to a grammar rule when longer words having higher confidence scores in the ASR result agree with those in the grammar rule. On the one hand, positive weights are given to the WFST output as rewards, when a word in the ASR result agree with the corresponding element in the grammar rule, in accordance with the ASR confidence and the length of the agreed words. On the other hand, negative weights are given as penalties when the words do not agree.

More specifically, we define reward $w_{ijk}$ as shown in Eq. (2). This is the case when a word is accepted and then becomes the $i$-th symbol of the WFST output; that is, word in an ASR result $e^{ASR}$ agrees with element $e^{gram(k)}$ in grammar rule $k$. Grammar element $e^{gram}$ is either a word or a slot.

$$w_{ijk} = CM(e^{ASR})l(e^{ASR}) \quad \text{[when the } i\text{-th output is } e^{ASR}\text{]}, \tag{2}$$

where $CM(e^{ASR})$ and $l(e^{ASR})$ denote the confidence score and the word length of $e^{ASR}$. The confidence score is between 0 and 1 [23]. The word length is calculated as the number of moras normalized by that of the longest word in the vocabulary.

Cases in which a word in the ASR result does not agree with an element in a grammar rule are regarded as errors, and the output symbol of the WFST becomes SUB, INS, or DEL. Output symbol SUB indicates that $e^{ASR}$ corresponds to grammar element $e^{gram(k)}$ but does not agree with it. In this case, penalty $w_{ijk}$ is defined as the average of the weights for $e^{ASR}$ and $e^{gram(k)}$.

$$\begin{aligned} w_{ijk} &= -\frac{1}{2}\{CM(e^{ASR})l(e^{ASR}) \\ &\qquad + CM(e^{gram(k)})l(e^{gram(k)})\} \\ &= -\frac{1}{2}\{CM(e^{ASR})l(e^{ASR}) + l(e^{gram(k)})\} \end{aligned} \tag{3}$$

[when the $i$-th output is SUB],

where $CM(e^{gram(k)}) = 1$ because there is no ASR confidence score for any grammar element. When $e^{gram(k)}$ is a slot in the grammar, the average word length in the slot is used instead of $l(e^{gram(k)})$.

Output symbol INS indicates that no element in grammar rule $k$ corresponds to $e^{ASR}$. In this case, $w_{ijk}$ is defined as the average length of words in the vocabulary, $\overline{l(e)}$, instead of the length of a grammar element, $l(e^{gram(k)})$, as is used in the definition of weights for SUB.

$$w_{ijk} = -\frac{1}{2}\{CM(e^{ASR})l(e^{ASR}) + \overline{l(e)}\} \tag{4}$$

[when the $i$-th output is INS]

Output symbol DEL indicates that no $e^{ASR}$ corresponds to grammar element $e^{gram(k)}$. In this case, $\overline{l(e)}$ is used instead of $l(e^{ASR})$ and $CM(e^{ASR}) = CM(e^{gram(k)}) = 1$ is assumed because there is no ASR result.

$$\begin{aligned} w_{ijk} &= -\frac{1}{2}\{CM(e^{ASR})l(e^{ASR}) \\ &\qquad + CM(e^{gram(k)})l(e^{gram(k)})\} \\ &= -\frac{1}{2}\{\overline{l(e)} + l(e^{gram(k)})\} \end{aligned} \tag{5}$$

[when the $i$-th output is DEL]

### 3.3 Example of Calculating Weights

Weights are calculated as illustrated in Fig. 4. In this example, the user utterance is, "Tell me a liaison of Koetsu-ji (a temple name)". The word "liaison" is not in the system vocabulary. The ASR result accordingly contains errors for that part; the result is, "Tell me all Sakyo-ward Koetsu-ji".

GV scores are calculated for each grammar rule that the system has. This example shows calculations for two grammar rules: [get_info] and [search_ward]. The former accepts "Tell me ⟨item name⟩ of ⟨temple name⟩", and the latter accepts "Tell me ⟨facility name⟩ of ⟨ward name⟩". Here, [] and ⟨⟩ denote a grammar rule and a slot in the grammar. This example also shows two WFST output sequences, which correspond to variable $j$ used in Sect. 3.2, for the single grammar rule [get_info].

We consequently obtain the result that grammar rule

User utterance: "Tell me a liaison of Koetsu-ji". (Underlined word is OOG.)

| ASR result | tell | me | all | Sakyo-ward (ward) | | Koetsu-ji (temple) | GV score |
|---|---|---|---|---|---|---|---|
| grammar rule [get_info] | tell | me | | ⟨item name⟩ | of | ⟨temple name⟩ | |
| WFST output | tell | me | INS | SUB | DEL | Koetsu-ji | |
| weights | +0.09 | +0.06 | −0.04 | −0.11 | −0.02 | +0.18 | +0.16 |
| grammar rule [get_info] | tell | me | ⟨item name ⟩ | of | | ⟨temple name⟩ | |
| WFST output | tell | me | SUB | SUB | | Koetsu-ji | |
| weights | +0.09 | +0.06 | −0.21 | −0.10 | | +0.18 | +0.02 |
| grammar rule [search_ward] | tell | me | | ⟨facility type⟩ | in | ⟨ward name⟩ | |
| WFST output | tell | me | INS | SUB | DEL | SUB | |
| weights | +0.09 | +0.06 | −0.04 | −0.12 | −0.02 | −0.21 | −0.24 |

**Fig. 4** Example of calculating weights in our grammar verification method.

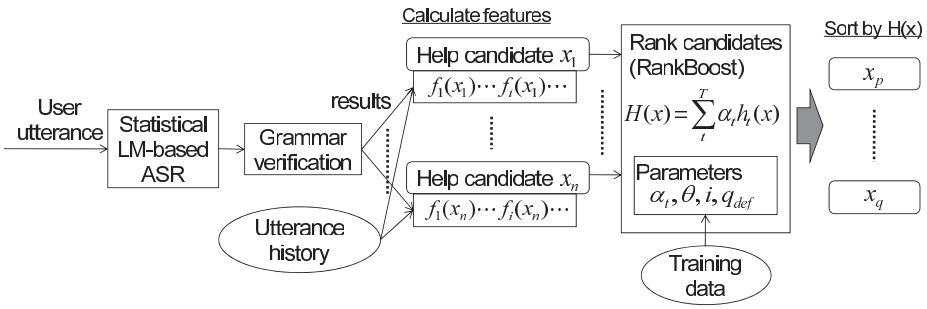



**Fig. 5** Outline of our method for ranking help message candidates.

**Table 1** Features used for ranking help message candidates.

| Feature | Description | Example value |
|---|---|---|
| **H1:** | GV score | 0.16 |
| **H2:** | GV score normalized by total GV score for all candidates | 0.09 |
| **H3:** | ratio of no. of accepted words in GV result to no. of all words | 0.5 |
| **H4:** | maximum no. of successively accepted words in GV result | 2 |
| **H5:** | no. of accepted slots in GV result | 0.13 |
| **H6:** | how much time before this grammar rule was selected as GV result (in no. of utterances) | 3 |
| **H7:** | maximum GV score for this grammar rule until current utterance | 0.40 |
| **H8:** | whether it belongs to "command" class | 1 |
| **H9:** | whether it belongs to "query" class | 0 |
| **H10:** | whether it belongs to "request-info" class | 0 |
| **H11–H17:** | products of H8 and each of H1 to H7 | (products of two values) |
| **H18–H24:** | products of H9 and each of H1 to H7 | (products of two values) |
| **H25–H31:** | products of H10 and each of H1 to H7 | (products of two values) |

GV: grammar verification

[get_info] has the highest score for this OOG utterance, and its GV score is +0.16. The result also indicates each error type as a result of the correspondence between the ASR result and the output sequence of the WFST: ⟨item name⟩ is replaced by "Sakyo-ward", "of" in the grammar rule [get_info] is deleted, and "all" in the ASR result is inserted.

## 4. Ranking Help Message Candidates by Using Utterance History

We use various types of information including the grammar verification (GV) result and the user's utterance history to rank the help message candidates. The latter helps make up for the information that is often absent from utterances or misrecognized in ASR results. Figure 5 outlines the method.

### 4.1 Features Used in Ranking

Table 1 lists the features used in our ranking method. They are calculated for the help message candidate corresponding to each grammar rule. Features H1 to H5 represent GV result reliability. H1 is the GV score given by Eq. (1). H2 is calculated by normalizing H1 by the total GV score for all grammar rules. This represents the GV result reliability relative to other GV results. H3 to H5 represent how well the user utterance matches the grammar rule.

Features H6 and H7 correspond to the utterance history. H6 reflects the case in which users tend to repeat similar utterances when their utterances are not understood by the

system. H7 represents whether and how well the user knows language expressions for the grammar rule. This feature corresponds to the *known degree* we previously proposed [5].

Features H8 to H10 represent properties of utterances corresponding to the grammar rules. They are categorized into three classes: "command", "query", and "request-info". The numbers of grammar rules in the three classes are 8, 4, and 11, respectively, in the sightseeing task. Here, "query" and "request-info" correspond to the two modes in the database search task: "specifying query conditions" and "requesting detailed information" [24]. In the model assumed here, the user first tries several query conditions to narrow down the entries and then formulates questions for individual entries found in the first mode. More specifically, the first mode deals with query conditions in the task, like "Please tell me sightseeing spots around ⟨location⟩"; the second mode deals with questions about specific entries in the target database such as "Please tell me how to get to ⟨temple name⟩". Therefore, utterances in either the "query" or "request-info" class tend to appear successively; in contrast, utterances in the "command" class tend to appear independently of the context. Features H11 to H31 are the products of features H8, H9, and H10 and each feature from H1 to H7. These were defined to take into account combinations of properties of utterances represented by H8, H9, and H10, and their reliability represented by H1 to H7, because RankBoost does not consider them automatically.

### 4.2 Ranking Algorithm

We choose to use RankBoost [6], a boosting algorithm based on machine learning. Help message candidates corresponding to each grammar rule are ranked as target instances $x$ of the algorithm. This algorithm is suitable because it can use training examples with multiple labels having a certain order of priority.

RankBoost arranges candidates $x$ in order by using a scoring function $H(x)$. Here, $H(x') < H(x'')$ means $x''$ is ranked higher than $x'$. This scoring function is defined as a linear combination of weak rankers giving partial information regarding the order:

$$H(x) = \sum_{t}^{T} \alpha_t h_t(x), \tag{6}$$

where $T$, $h_t()$, and $\alpha_t$ denote the number of boosting iterations, a weak ranker, and its associated weight, respectively. Weak ranker $h_t$ is defined by comparing the value of feature $f_i$ of candidate $x$ with threshold $\theta$. That is,

$$h_t(x) = \begin{cases} 1 \text{ if } f_i(x) > \theta \\ 0 \text{ if } f_i(x) \leq \theta \\ q_{def} \text{ if } f_i(x) = \perp \end{cases}, \tag{7}$$

where $q_{def} \in \{0, 1\}$. Here, $f_i(x)$ denotes the value of the $i$-th feature of candidate $x$, and $\perp$ denotes that no value is given in $f_i(x)$.

## 5. Experimental Evaluation

### 5.1 Target Data

We collected utterance data from 30 participants by using a multi-domain spoken dialogue system that handles five domains: restaurant, hotel, sightseeing, bus, and weather [25]. The data were collected under the following conditions: the participants were given no instructions on the language expressions that the system accepts. System responses were given by a text-to-speech (TTS) system without a display screen. The participants were given six scenarios describing the tasks to be completed.

The data consisted of 180 dialogues and 11,733 utterances. Data from five participants were used to determine the number of boosting iterations and to improve the language models (LMs) used for ASR. We used utterances in the restaurant, hotel, and sightseeing domains because the remaining two, bus and weather, did not have many grammar rules. The number of grammar rules in the bus and weather domains was 12 and 11. Out of these, 8 belonged to the "command" class. We then extracted utterances including many OOG ones on the basis of the GV scores to evaluate the performance of our method for such utterances. Specifically, we collected utterances that had negative GV scores, calculated using Eq. (1). Many OOG utterances were included, although a negative GV score does not necessarily indicate that the utterance is OOG. As a result, 1,349 utterances by 25 participants were used as the evaluation data. They consisted of 363 utterances in the restaurant domain, 563 in the hotel domain, and 423 in the sightseeing domain.

We used Julius[†] as the ASR engine. We constructed class 3-gram LMs for ASR by using 10,000 sentences generated from the task grammars and 600 utterances collected from the five participants. The vocabulary sizes for the restaurant, hotel, and sightseeing domains were 3,456, 2,625, and 3,593, and their ASR accuracies were 45.8%, 57.1%, and 43.5%, respectively. These ASR accuracies were not high because the target utterances included many OOG utterances.

To confirm that many OOG utterances were included in these utterances, we also measured the utterance accuracies of ASR using the grammar-based ASR engine Julian with LMs that were the same grammar set used in the grammar FSTs. The accuracies approximately correspond to the performance when we simply use an FST-based parser. The accuracies were 11.6%, 7.3%, and 7.1%, respectively. This means the performance of a conventional FST parser would also be very low for the OOG utterances. Note that the grammar did not have very wide coverage for the target utterances. This was because all of the participants were novices at using the system and were not given any instruction about which language expressions were accepted by the system. Another reason was that we did not add grammar

[†]http://julius.sourceforge.jp/

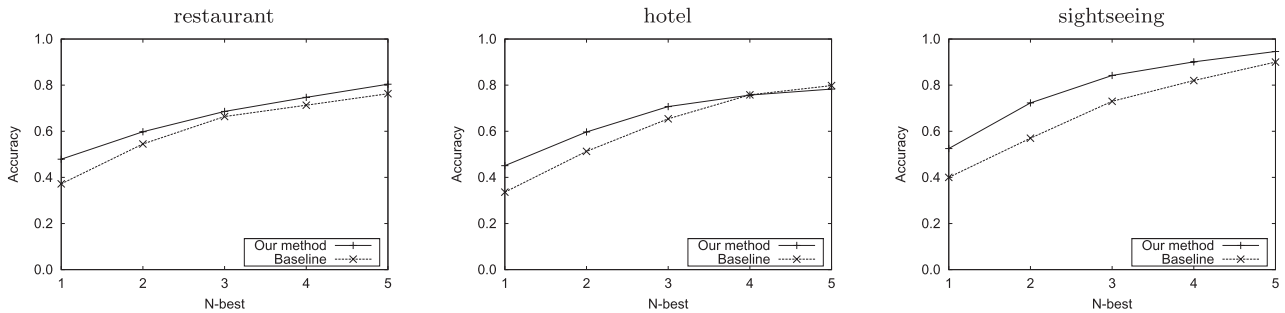



**Fig. 6** Accuracy when $N$ candidates were provided. ($1 \leq N \leq 5$)

rules after collecting the utterances in order to fix the definition of OOG utterances used here.

The set of possible thresholds in the weak rankers described in Sect. 4.2 consisted of all feature values that appeared in the training data. The number of boosting iterations was determined on the basis of the accuracy of the data collected from the five participants: 400, 100, and 500 for the restaurant, hotel, and sightseeing domains.

### 5.2 Reference Data and Evaluation Criterion

We manually selected five help messages corresponding to grammar rules as reference labels per utterance in the order of having the strongest to weakest relation to the utterance. We gave such multiple labels because the most appropriate help message cannot necessarily be determined uniquely, especially when the user utterance is short and does not contain sufficient information. The labels were given by looking at not only the target utterance but also the preceding utterances as context information. RankBoost was trained by using the order among the five reference labels per utterance; the order among the other candidates except the five reference labels in the utterance was not used for the training. There were 28, 27, and 23 grammar rules in the restaurant, hotel, and sightseeing domains, respectively†. That is, the number of help message candidates to be ranked was 28 in the restaurant domain and so forth. When an amount of training data is small, a grammar rule that is not selected as a reference label in the training set can appear as a reference label in a test set. Such a rule is possible to be ranked correctly if its feature values are similar with those ranked higher in the training set; this is because relationships in the training data are modeled not as grammar rules themselves, but as instances having certain feature values.

Several figures showing how the reference labels distributed in the evaluation data are listed in Table 2. Here, we define relative frequency as occurrences of each label divided by their total occurrences as reference labels in the evaluation data. Entropy is calculated by regarding this frequency as probability. From the table, we can see that grammar rules to be ranked were not relatively varied in the sightseeing domain because there were less grammar rules and its entropy was lower than the other two. Those of the hotel and restaurant domains were almost the same.

**Table 2** Distribution of reference labels $x$.

| | restaurant | hotel | sightseeing |
|---|---|---|---|
| No. of grammar rules ($n$) | 26 | 26 | 23 |
| Max. relative frequency | 0.093 | 0.100 | 0.113 |
| Min. relative frequency | 0.003 | 0.002 | 0.0004 |
| Ave. relative frequency | 0.0385 | 0.0385 | 0.0435 |
| Sample stdev. | 0.0261 | 0.0287 | 0.0392 |
| Entropy $H(x)$ | 4.380 | 4.324 | 3.935 |
| $H(x)/\log_2(n)$ | 0.932 | 0.920 | 0.870 |

stdev.: standard deviation

As a metric, we used the ratio of the number of utterances with at least one of the reference labels in the top $N$ candidates. This corresponds to the probability that at least one appropriate help message was contained in a list of $N$ candidates. The accuracy was calculated by 5-fold cross validation. As the baseline method, the help messages were provided using only the GV scores.

### 5.3 Results

Figure 6 plots the results for the three domains: restaurant, hotel, and sightseeing. On the whole, accuracies in the sightseeing domain were higher than those for the other two. This corresponds to the fact listed in Table 2, that is, the entropy in the sightseeing domain was lower than those of the restaurant and hotel domains. The accuracies in the two domains were similar as well as their entropy listed in the table.

The average differences between the baseline and our method for the three domains were 11.7 points for $N = 1$, 9.7 points for $N = 2$, and 6.7 points for $N = 3$; i.e., the differences were larger, the smaller the $N$. The following differences in accuracy were statistically significant ($p < 0.05$) in a McNemar test: $N = 1$ in the restaurant domain, $N = 1$, 2, 3 in the hotel domain, and $N = 1$, 2, 3, 4, 5 in the sightseeing domain. These results indicate that we can reduce the number of help messages when several messages are provided to the user. The improvements result from the features we incorporated, such as the estimated user knowledge, in addition to the GV results.

We also identified the dominant features by summing the absolute values of final weight $\alpha$ for each feature in

†Out of these, two and one grammar rules in the restaurant and hotel domains did not appear as reference labels in our evaluation data.

**Table 3** Sum of absolute values of weight $\alpha$ for features.

| H7 | H17 (H7*H8) | H19 (H2*H9) | H2 | H6 |
|---|---|---|---|---|
| 9.58 | 6.91 | 6.61 | 6.02 | 6.01 |

RankBoost. The five dominant features are shown in Table 3. They include the feature obtained from the GV result (H2), the feature reflecting the user's utterance history (H6), the feature representing the estimated user knowledge (H7), and the features representing properties of the utterance (H8, H9). The most dominant feature was H7, which appears twice in the table. This feature uses the GV scores and if the score is high, an utterance is assumed to be correctly accepted by the system. After such an utterance, it can be assumed that the user already knows the language expression and its grammar rule, so a help message corresponding to the grammar rule is unnecessary and should not be provided. This feature improved the performance of the ranking by giving a lower rank to help messages that previously had high maximum GV scores. The second most dominant feature was H2, which shows that using the GV result is effective.

## 6. Conclusion

We addressed the problem of OOG utterances in spoken dialogue systems and developed a method for generating help messages. Each help message corresponds to a grammar rule in the system, and the grammar rule that the user intended to use is identified on the basis of the GV score. The help message candidates are ranked on the basis of the user's utterance history and the grammar verification results.

The evaluation described here was based on utterances collected beforehand. Another user study is required to evaluate how effective providing such help messages is in real dialogues. Integration with other dialogue strategies such as generating clarification questions and asking for values of unfilled slots in a grammar rule identified by grammar verification should also be investigated. We used RankBoost, one of several ranking algorithms. Other ranking algorithms, such as Ranking SVM [26], may achieve better performance; this needs to be confirmed experimentally. Finally, we assumed that the example language expressions in the help messages were fixed. We need to investigate what kinds of expressions would be most helpful in guiding novice users.

## Acknowledgments

This work was partially supported by KAKENHI and SCAT.

### References

[1] K. Komatani, S. Ikeda, Y. Fukubayashi, T. Ogata, and H.G. Okuno, "Ranking help message candidates based on robust grammar verification results and utterance history in spoken dialogue systems," Proc. SIGDIAL Conference, pp.314–321, 2009.

[2] A. Raux, D. Bohus, B. Langner, A.W. Black, and M. Eskenazi, "Doing research on a deployed spoken dialogue system: One year of Let's Go! experience," Proc. Int'l Conf. Spoken Language Processing (INTERSPEECH), 2006.

[3] K. Komatani, T. Kawahara, and H.G. Okuno, "Analyzing temporal transition of real user's behaviors in a spoken dialogue system," Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp.142–145, 2007.

[4] R. Nisimura, A. Lee, M. Yamada, and K. Shikano, "Operating a public spoken guidance system in real environment," Proc. European Conf. Speech Commun. & Tech. (EUROSPEECH), pp.845–848, 2005.

[5] Y. Fukubayashi, K. Komatani, T. Ogata, and H.G. Okuno, "Dynamic help generation by estimating user's mental model in spoken dialogue systems," Proc. Int'l Conf. Spoken Language Processing (INTERSPEECH), pp.1946–1949, 2006.

[6] Y. Freund, R.D. Iyer, R.E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," Journal of Machine Learning Research, vol.4, pp.933–969, 2003.

[7] M. Dzikovska, C. Callaway, E. Farrow, J. Moore, N. Steinhauser, and G. Campbell, "Dealing with interpretation errors in tutorial dialogue," Proc. SIGDIAL Conference, pp.38–45, 2009.

[8] D. Bohus and A.I. Rudnicky, "Sorry, I didn't catch that! — An investigation of non-understanding errors and recovery strategies," Proc. 6th SIGdial Workshop on Discourse and Dialogue, pp.128–143, 2005.

[9] C.A. Kamm, D.J. Litman, and M.A. Walker, "From novice to expert: The effect of tutorials on user expertise with spoken dialogue systems," Proc. Int'l Conf. Spoken Language Processing (ICSLP), pp.1211–1214, 1998.

[10] S. Tomko, T.K. Harris, A. Toth, J. Sanders, A. Rudnicky, and R. Rosenfeld, "Towards efficient human machine speech communication: The Speech Graffiti project," ACM Trans. Speech Lang. Process., vol.2, no.1, 2005.

[11] S. Tomko and R. Rosenfeld, "Shaping user input in speech graffiti: A first pass," CHI '06 Extended Abstracts on Human Factors in Computing Systems, pp.1439–1444, 2006.

[12] G. Gorrell, I. Lewin, and M. Rayner, "Adding intelligent help to mixed-initiative spoken dialogue systems," Proc. Int'l Conf. Spoken Language Processing (ICSLP), pp.2065–2068, 2002.

[13] B.A. Hockey, O. Lemon, E. Campana, L. Hiatt, G. Aist, J. Hieronymus, A. Gruenstein, and J. Dowding, "Targeted help for spoken dialogue systems: Intelligent feedback improves naive users' performance," Proc. 10th Conf. of the European Chapter of the ACL (EACL2003), pp.147–154, 2003.

[14] C. Lee, S. Jung, D. Lee, and G.G. Lee, "Example-based error recovery strategy for spoken dialog system," Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp.538–543, 2007.

[15] A. Gruenstein and S. Seneff, "Releasing a multimodal dialogue system into the wild: User support mechanisms," Proc. 8th SIGdial Workshop on Discourse and Dialogue, pp.111–119, 2007.

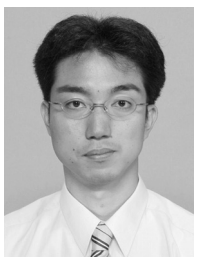
[16] M. Hartmann and D. Schreiber, "Proactively adapting interfaces to individual users for mobile devices," Adaptive Hypermedia and Adaptive Web-Based Systems, 5th International Conference (AH 2008), Lecture Notes in Computer Science, LNCS 5149, pp.300–303, Springer, 2008.

[17] E. Ammicht, A. Potamianos, and E. Fosler-Lussier, "Ambiguity representation and resolution in spoken dialogue systems," Proc. European Conf. Speech Commun. & Tech. (EUROSPEECH), pp.2217–2220, 2001.

[18] J.D. Williams, "Using particle filters to track dialogue state," Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp.502–507, 2007.

[19] R. Higashinaka and M. Nakano, "Ranking multiple dialogue states by corpus statistics to improve discourse understanding in spoken dialogue systems," IEICE Trans. Inf. & Syst., vol.E92-D, no.9,

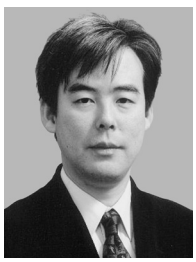
pp.1771–1782, Sept. 2009.
[20] S.S. Pradhan and W.H. Ward, "Estimating semantic confidence for spoken dialog systems," Proc. IEEE Int'l Conf. Acoust., Speech & Signal Processing (ICASSP), pp.233–236, 2002.
[21] R. Higashinaka, M. Nakano, and K. Aikawa, "Corpus-based discourse understanding in spoken dialogue systems," Proc. Annual Meeting of the Association for Computational Linguistics (ACL), pp.240–247, 2003.
[22] Y. Fukubayashi, K. Komatani, M. Nakano, K. Funakoshi, H. Tsujino, T. Ogata, and H.G. Okuno, "Rapid prototyping of robust language understanding modules for spoken dialogue systems," Proc. International Joint Conference on Natural Language Processing (IJCNLP), pp.210–216, 2008.
[23] A. Lee, K. Shikano, and T. Kawahara, "Real-time word confidence scoring using local posterior probabilities on tree trellis search," Proc. IEEE Int'l Conf. Acoust., Speech & Signal Processing (ICASSP), pp.793–796, 2004.
[24] K. Komatani, N. Kanda, T. Ogata, and H.G. Okuno, "Contextual constraints based on dialogue models in database search task for spoken dialogue systems," Proc. European Conf. Speech Commun. & Tech. (EUROSPEECH), pp.877–880, 2005.
[25] K. Komatani, S. Ikeda, T. Ogata, and H.G. Okuno, "Managing out-of-grammar utterances by topic estimation with domain extensibility in multi-domain spoken dialogue systems," Speech Commun., vol.50, no.10, pp.863–870, 2008.
[26] T. Joachims, "Optimizing search engines using clickthrough data," Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp.133–142, 2002.



**Kazunori Komatani** received B.E., M.S., and Ph.D. degrees in Informatics in 1998, 2000, 2002, from Kyoto University, Japan. In 2002, he became an Assistant Professor in the Graduate School of Informatics, Kyoto University. He is currently an Associate Professor in the Graduate School of Engineering, Nagoya University. From 2008 to 2009, he was a Visiting Scientist at Carnegie Mellon University, Pittsburgh, PA, USA. He has received several awards including the 2002 FIT Young Researcher Award and the 2004 IPSJ Yamashita SIG Research Award, both from the Information Processing Society of Japan (IPSJ). His research interests center on spoken language processing, especially on spoken dialogue systems. He is a member of the IPSJ, NLP, JSAI, ACL, and ISCA.

**Yuichiro Fukubayashi** received B.E. and M.S. degrees in Informatics in 2006 and 2008 from Kyoto University, Japan. He currently works at NEC Corporation.

**Satoshi Ikeda** received B.E. and M.S. degrees in Informatics in 2007 and 2009 from Kyoto University, Japan. He currently works at Canon Inc.



**Tetsuya Ogata** received the B.S., M.S. and D.E. degrees in Mechanical Engineering in 1993, 1995, and 2000, respectively, from Waseda University. From 1999 to 2001, he was a Research Associate in Waseda University. From 2001 to 2003, he was a Research Scientist in the Brain Science Institute, RIKEN. Since 2003, he has been a Faculty Member in the Graduate School of Informatics, Kyoto University, where he is currently an Associate Professor. He received the 2000 JSME Outstanding Paper Medal from the Japan Society of Mechanical Engineers, and the Best Paper Award of IEA/AIE-2005. He is a member of the IPSJ, JSAI, RSJ, HIS, SICE, and IEEE.

**Hiroshi G. Okuno** received the B.A. and Ph.D. degrees from the University of Tokyo, Japan, in 1972 and 1996, respectively. He is currently a Professor of the Graduate School of Informatics, Kyoto University, Japan. He received various awards including the Best Paper Awards of JSAI. His research interests include computational auditory scene analysis, robot audition and music scene analysis.