Contextual Constraints based on Dialogue Models in Database Search Task for Spoken Dialogue Systems

Kazunori Komatani, Naoyuki Kanda, Tetsuya Ogata, Hiroshi G. Okuno

Graduate School of Informatics, Kyoto University
Yoshida-Hommachi, Sakyo, Kyoto 606-8501, Japan
komatani@i.kyoto-u.ac.jp

Abstract

This paper describes the incorporation of contextual information into spoken dialogue systems in the database search task. Appropriate dialogue modeling is required to manage automatic speech recognition (ASR) errors using dialogue-level information. We define two dialogue models: a model for dialogue flow and a model of structured dialogue history. The model for dialogue flow assumes dialogues in the database search task consist of only two modes. In the structured dialogue history model, query conditions are maintained as a tree structure, taking into consideration their inputted order. The constraints derived from these models are integrated by using a decision tree learning, so that the system can determine a dialogue act of the utterance and whether each content word should be accepted or rejected, even when it contains ASR errors. The experimental result showed that our method could interpret content words better than conventional one without the contextual information. Furthermore, it was also shown that our method was domain-independent because it achieved equivalent accuracy in another domain without any more training.

1. Introduction

Automatic speech recognition (ASR) errors are inherent to speech interface. Appropriate interaction such as confirmations or disambiguating questions can keep spoken dialogue systems from fatal failure caused by such errors. In generating efficient confirmation, confidence measures of ASR were effective [1], which were derived from single utterances. We address to detect ASR errors not only by the utterance itself but also by using contextual information based on each situation together.

Dialogue acts were often designed to formulate contextual information for spoken dialogue systems, so that the connection probability between them was used as a contextual constraint. Generally speaking, if dialogue acts were designed more minutely, more useful constraints could be obtained. However, at the same time, it means that more data would be required to train the statistical model (N-gram model [2, 3]) and portability would also be spoiled. It is inherently difficult to collect a sufficient amount of dialogue data, because data units are larger than words and complicated dialogue acts are automatically annotated only with difficulty. Therefore, to incorporate contextual information into spoken dialogue systems, dialogue modeling that does not depend on specific domains and is informative enough to screen out errors is required.

We direct our attention to an abstract structure of a database search task, and model it as only two modes: "specifying query conditions" and "requesting detailed information". Then, we define a set of very simple dialogue acts corresponding to the above dialogue model. Furthermore, we create a model to maintain query conditions as a tree structure, which can be used as a weight between attributes of query conditions. By integrating information derived from these models using a decision tree, we address obtaining useful constraints to interpret ASR results in a database search task without depending on any specific domains.

2. Models in Database Search Task

Araki et al. [4] categorized dialogues in task-oriented spoken dialogue systems into three abstract tasks based on the direction of information flow. The abstract tasks consist of slot filling, database search, and explanation. Our task corresponds to the second one, in which information necessary to achieve the task is offered to each other. Spoken dialogue systems that retrieve information from a relational database, which is one of the most popular back-ends, can be categorized here. In this abstract task, users often determine their next utterance according to the system's preceding response, because users cannot know whether (or how many) there are entries satisfying their request in the target database until the system tells them. The characteristics of this abstract task are summarized as follows.

- Requisite attributes are different from users
- User's final goal may change according to intermediate results of queries.

According to these interactive properties, it is impossible to describe a dialogue procedure beforehand as a system-initiated dialogue, so we need to adopt mixed-initiated one.

The target database of this abstract task is an ordinary relational database, which has attribute-value pairs for each entry and has a key attribute whose value is unique in the database. Hereafter, we set a restaurant database as a query target. The key attribute in this database is the restaurant name. Note that our method models the database search task, but never depends on its domain, that is, content of the database.

2.1. Model for Dialogue Flow

Our modeling of a typical dialogue flow in the database search task follows. First, users gradually specify (or retract) query conditions they have thought of, and narrow down entries that meet their request[1], into several ones[2]. Then, users put some questions for the individual entries specified in the preceding interactions, and obtain information from the database. We

[1]This process cannot be realized by a system-initiated manner, because the order of preferred attributes is different from users.

[2]We assume this number of entries should be a limit the system can read out.

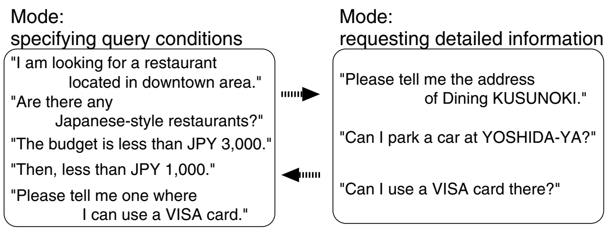
Mode:
specifying query conditions
"I am looking for a restaurant located in downtown area."
"Are there any Japanese-style restaurants?"
"The budget is less than JPY 3,000."
"Then, less than JPY 1,000."
"Please tell me one where I can use a VISA card."
Mode:
requesting detailed information
"Please tell me the address of Dining KUSUNOKI."
"Can I park a car at YOSHIDA-YA?"
"Can I use a VISA card there?"

Figure 1: Two modes in model for dialogue flow

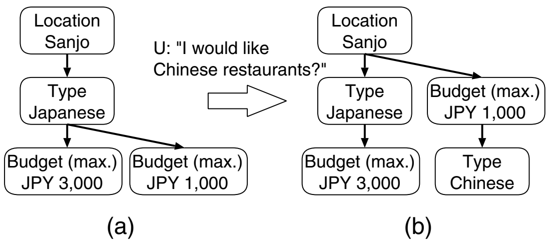

Figure 2: Example of structured dialogue history

broadly model the former as a "specifying query conditions" mode, and the latter as a "requesting detailed information" mode. Utterances are categorized under one of the two modes, and the dialogues proceed by transiting between the two modes (Figure 1). For example, "I am looking for restaurants at a downtown area." corresponds to the "specifying query conditions" mode, and "Please tell me the address of that restaurant." corresponds to the "requesting detailed information" mode. As shown in this example, expressions appearing in each mode are different, because the former includes those corresponding to query conditions whereas the latter is a question about specific entries. This model is helpful when the system determines speech acts of utterances, rejects content words unrelated to the current context, and so on.

2.2. Model of Structured Dialogue History

We hold a tree-structured dialogue history, whose nodes consist of query conditions (attribute-value pairs). The query conditions are maintained to satisfy the following constraints: a more recently input condition is put at a lower position on the tree, and when a condition having the same attribute is added, a new node is once generated to the position of the sister of the node. We can represent the following degrees using this model: a change in the upper part of the tree means larger conversion of the current topic, and a parent node having many descendants has not been changed for a long time, and accordingly is important. These degrees are used as contextual constraints.

Figure 2 shows an example of manipulating the structured dialogue history. The left-hand part (a) in this figure represents a state when query condition inputted in the following order: [location: Sanjo], [type: Japanese], [budget (max.): JPY 3,000], and [budget (max.): JPY 1,000]. Current query conditions can be derived by traversing the right-most descendants. If [type: Chinese] is inputted for the state (a) in Figure 2, [type: Chinese] is once added to the position of the younger sister of [type: Japanese]. On this occasion, [budget (max.): JPY 1,000] moves to the higher position than the [type: Chinese] based on the constraint. As a result, the right-hand tree (b) in Figure 2 is derived.

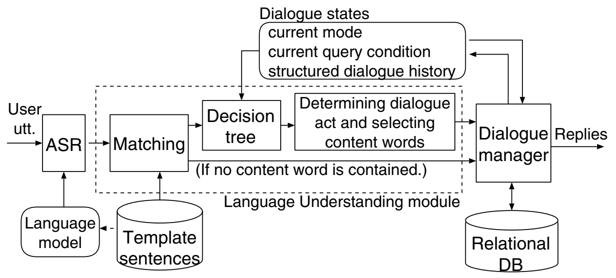

Figure 3: Overview of our system for database search task

3. Language Understanding using Contextual Constraint

We introduce contextual constraints based on our models. An overview of our system is described in Figure 3. The system mainly consists of an automatic speech recognizer (ASR), a language-understanding module, and a dialogue manager. The language-understanding module determines a dialogue act and whether content words of utterances should be accepted or not, using both ASR outputs and context. A dialogue manager changes dialogue states using the results of the language understanding, and generates responses by accessing a database based on the states. We will describe details of the language-understanding module in the following sections.

3.1. Matching using Template Sentences

We calculate similarities between an ASR result and template sentences, which were manually assigned to each dialogue act beforehand. These similarities represent how an ASR result as a sentence resembles the prepared sentences having each dialogue act. The template sentences amount to 590, for example, "Please tell me `<TYPE>`-style restaurants". Here, the `<TYPE>` denotes values for `<TYPE>` attribute in the target database.

We treat 7 dialogue acts in our language-understanding module. Two dialogue acts such as `specify conditions` and `request information` correspond to the two modes in the model of dialogue flow: "specifying query conditions" and "requesting detailed information", which constitute the main part of dialogues in the database search task. We add 5 supplemental dialogue acts, such as `delete conditions`, `yes`, `no`, `clear all conditions`, and `undo`. The content words are defined as all attribute names and their values in the target relational database.

If the similarity for either `yes`, `no`, `clear all condition`, or `undo`, which contain no content word, is the highest, a dialogue act for the utterance is determined as them. Otherwise, either `specify conditions`, `request information`, or `delete condition` get the highest similarity, which include content words, the dialogue act is determined by the procedures described in the following sections.

3.2. Construction of Decision Tree Incorporating Contextual Constraint

We constructed a decision tree [5] that outputs likelihoods which dialogue act a word should be assigned to or whether a word should be rejected, for each content word. In the training phrase, correct dialogue acts were given to each content word by hand, and a `reject` label was given to speech recognition errors. The influence due to recognition errors can be avoided

S1	dialogue act having the highest likelihood
S2	value of likelihood of **S1**
S3	value of likelihood of the second-best dialogue act
S4	**(S2) / (S3)**
S5	kind of content word (attribute / value / key attribute / value of key attribute)
S6	confidence measure of ASR
S7	existence of key-attribute pair (e.g., 'budget' and 'JPY 1,000' in a single utterance)

Figure 4: Features obtained from single utterance

if those are correctly classified as `reject`.

We use two groups of features in classification: **features obtained from a single utterance** listed in Figure 4, and **features based on the proposed models** listed in Figure 5. As features obtained from a single utterance, we adopt the results of the matching (S1, S2, S3, S4) and a confidence measure of ASR (S6). On the other hand, modes of the current utterance (C1) and the number of entries (restaurants in this domain) that agree with the current query condition (C2) are based on the dialogue flow model. C8 denotes the rate of entries that have ever been checked since the current mode became "requesting detailed information"[3]. The depth of a node having the same attribute as the current utterance and the number of descendant nodes of it (C9) are based on the model of structured dialogue history. The depth and its average were normalized by either ratio or difference of the current depth of the whole tree, respectively (C16, C17, C18, C19).

3.3. Determining Dialogue Act and Selecting Content Words

The language-understanding module determines the dialogue act as a whole utterance by integrating every output of the decision tree for content words. Then, it selects each content word based on both the determined dialogue act and the likelihood that the word should be rejected.

First, likelihoods $CF(s|F_i, w_j)$ of a dialogue act s for each content word w_j are obtained by the decision tree[4]. F_i denotes current situations in the dialogue, which are represented by features for the decision tree. The likelihoods are summed up for every content word, and the dialogue act S_i having the largest sum total is determined as that of the utterance i. Here, $s \in \{$`specify conditions`, `require information`, `delete condition`$\}$, which have content words.

$$S_i = \arg\max_s \sum_j CF(s|F_i, w_j)$$

Then, we also calculate likelihoods $R_j = CF(\texttt{reject}|F_i, w_j)$, which represents the word should be rejected, for each content word w_j. Then, each content word is accepted if the likelihood of the determined dialogue act $CF(S_i|F_i, w_j)$ is greater than R_j. Otherwise, it is rejected.

[3]This feature is designed to represent a situation where the mode is apt to return to "specifying query conditions" after most restaurants in current conditions have already checked by the user.

[4]$CF(s|F_i, w_j)$ is outputted by the decision tree, which smoothes out accuracies in training data. When w_j is classified to a leaf in the decision tree, $CF(s|F_i, w_j)$ is given as $(M + 1)/(N + P)$, where N is the number of elements at the leaf, P is the number of dialogue acts, and M is the number of samples with s in training data [5].

C1	current mode
C2	# entries that agree with current query condition
C3	# entries that agree with current query condition and have ever been checked since entering current mode
C4	# entries that have ever been checked
C5	depth of current tree
C6	dialogue act at previous utterance
C7	whether preceding system's prompt is question or not
C8	**(C3) / (C2)**
C9	# of descendants of node having same attribute as input value
C10	whether there is same one in current query condition if input word is either value or attribute
C11	whether it has ever been checked in dialogue if input word is value of key attribute
C12	whether it agrees with current query condition if input word is value of key attribute
C13	whether it has been confirmed
C14	whether it has been denied
C15	whether it has been deleted
C16	(depth of same attribute as input) / ((**C5**) + 1)
C17	((**C5**) + 1) − (depth of same attribute as input)
C18	(ave. depth of same attribute as input) / ((**C5**) + 1)
C19	((**C5**) + 1) − (ave. depth of same attribute as input)
C20	whether # entries agreeing with current query condition is 0
C21	whether # entries agreeing with current query condition is 1

Figure 5: Features based on proposed models

4. Experimental Evaluation

4.1. Implementation and Data Collection

We constructed a spoken dialogue system retrieving a restaurant database, which has 1,217 entries. The key attribute is the restaurant name, and there are 12 other attributes, such as type, address, and so on. We adopted a speech recognizer Julius[5] [6], which works based on statistic language models (LMs). The vocabulary size of the LM was 21,565.

Our system outputs results both in text on a console and by a TTS. When more than 8 entries are obtained, the system outputs the number of restaurants that match the query conditions. Otherwise, the restaurants' names and numbers are outputted.

We collected dialogue data from 20 novice subjects who had not used spoken dialogue systems. First, we gave a brief instruction about the system and showed some example sentences the system could accept. Then, subjects were asked to find a few appropriate restaurants for several scenarios; for example, "You want to eat some Japanese food, and you only have a VISA card. Please find a few suitable restaurants." If subjects found appropriate restaurants that met the given situations based on their criteria, they could finish the dialogue. Each subject used the system 4 times.

We obtained 3,015 utterances for 20 subjects (151 utterances per subject; 38 utterances per dialogue), which contained 2,948 content words in total. The average content word accuracy of the ASR was 81.9%. There were 389 content word recognized incorrectly, which should be judged as `reject` by the language-understanding module.

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

Table 1: Classification accuracy of language understanding in restaurant system

	#1	#2	#3
Specify condition	0.926	0.907	0.903
Request information	0.945	0.953	0.949
Delete conditions	0.815	0.730	0.857
Reject	0.100	0.368	0.550
Total	0.809	0.834	0.867

Table 2: Classification accuracy of language understanding in hotel system

	#1	#2	#3	#4
Specify condition	0.917	0.940	0.964	0.930
Request information	0.978	0.967	0.983	0.990
Delete conditions	0.822	0.711	0.756	0.933
Reject	0.287	0.318	0.504	0.527
Total	0.888	0.890	0.926	0.924

4.2. Evaluation of Language Understanding Module

We compared the following three methods to evaluate our language-understanding module with the proposed models.

Method #1 A baseline method, in which a dialogue act having the highest likelihood in the matching was simply adopted, and content words whose confidence measures of the ASR were less than a threshold were rejected. No decision tree learning was used.

Method #2 Decision tree learning based on the proposed method without contextual features shown in Figure 5.

Method #3 The proposed method with all features.

In method #1, we had optimized the threshold for the confidence measure of the ASR, and set it to 0.05, which minimized errors. In methods #2 and #3, the experiments were carried out with subject-open 10-fold cross validation, that is, utterances of 2 subjects out of 20 were used as the test data, and the remainder was used as the training data. The process was repeated 10 times, and the average of the accuracies was computed. We also optimized the pruning parameter in constructing a decision tree to minimize errors. The classification accuracy of the language-understanding module per content word is shown in Table 1.

In method #3, the accuracy was improved by 3.3%, which was owed to the incorporation of contextual features. This improvement was notable in `reject`. Although the first split of the decision tree in method #3 was made by "dialogue act having the highest likelihood" as results of the matching, features derived from the proposed dialogue model played an important role in the remaining part of the tree. For classifying content words corresponding to values in relational databases, these were C1 and features based on the structured dialogue history (C16-C19). For either attributes or values corresponding to the key attribute, C2, C11, and C12 were effective.

4.3. Application to Another Domain

We also apply our method to a system retrieving a hotel database. The number of entries was 2,004. The key attribute was the hotel name, and there were 6 other attributes, such as type, location, equipment, and so on. We also collected dialogue data from 10 subjects, and obtained 1,271 utterances, which contained 1,426 content words. The average word accuracy of the ASR was 85.8%. We added another condition to verify domain-independency.

Method #4 Decision tree trained with data in another domain was used. Here, decision tree was trained with the restaurant data, and was evaluated by the hotel data.

The accuracy of the language-understanding module for the hotel domain is shown in Table 2.

Our method also improved the accuracy in this domain, which can be seen as the difference between method #1 and #3. Furthermore, accuracy in method #4 was nearly equivalent to that in method #3, in which both the training and the test data were in the same domain. This means contextual constraint derived from the decision tree can be applied to other domains.

4.4. Discussion

In the evaluation for the restaurant system (Table 1), there were still 373 interpretation errors (content words which had interpreted as incorrect dialogue acts). Note that, out of 2,803 recognized content words, 389 words had been ASR errors. This fact meant that the system managed to interpret utterances even though they contained errors. The most frequent interpretation errors were words incorrectly classified as `specifying conditions`, which should be `reject`. This kind of the error amounted to 133 words. One of causes of these errors was because sufficient constraints cannot be obtained in initial parts of dialogues.

5. Conclusion

We incorporated contextual information by defining new models in the database search task. The models are not dependent on the specific domain. Dialogue modeling that does not depend on specific domains and is informative enough to screen out errors is required to complement statistical information because of inherent difficulty in collecting sufficient amounts of dialogue data. We showed an example of dialogue modeling that works as useful constraints.

6. References

[1] T. J. Hazen, T. Burianek, J. Polifroni, and S. Seneff, "Integrating recognition confidence scoring with language understanding and dialogue modeling," in *Proc. ICSLP*, 2000.

[2] R. Higashinaka, M. Nakano, and K. Aikawa, "Corpus-based discourse understanding in spoken dialogue systems," in *Proc. ACL*, 2003, pp. 240–247.

[3] N. Reithinger and E. Maier, "Utilizing stastical dialog act processing in verbmobil," in *Proc. ACL*, 1995, pp. 116–121.

[4] M. Araki, K. Komatani, T. Hirata, and S. Doshita, "A dialogue library for task-oriented spoken dialogue systems," in *Proc. IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 1999, pp. 1–7.

[5] J. R. Quinlan, *C4.5: Programs for Machine Learning.*, Morgan Kaufmann, San Mateo, CA, 1993, http://www.rulequest.com/see5-info.html.

[6] T. Kawahara, A. Lee, K. Takeda, K. Itou, and K. Shikano, "Recent progress of open-source LVCSR engine Julius and Japanese model repository," in *Proc. ICSLP*, 2004, pp. 3069–3072.