Dialogue Helpsystem based on Flexible Matching of User Query with Natural Language Knowledge Base

Sadao Kurohashi and Wataru Higasa
Graduate School of Informatics
Kyoto University
Yoshida-honmachi, Sakyo, Kyoto, 606-8501 Japan
kuro@i.kyoto-u.ac.jp, higasa@pine.kuee.kyoto-u.ac.jp

Abstract

This paper describes a dialog helpsystem which advises users in using computer facilities and software applications provided by the Center for Information and Multimedia Studies, Kyoto University. The system employs a knowledge base written in natural language and retrieves a proper knowledge unit by flexible matching of user query with the knowledge base. The system is running since July 1999, received about 2,000 queries for the first seven months, and answered about 40% of them satisfactory.

1 Introduction

One of the ultimate goals of Natural Language Processing is to realize a dialogue system which can communicate with human beings in a natural way (Wilks, 1999). However, no effective real world dialogue application exists so far, not only in spoken language for which speech recognition is still a big obstacle, but also in typing language.

The most serious problem is knowledge. It is obvious that without sufficient knowledge a dialogue system cannot talk with people satisfactorily. Classical dialogue systems like UC (Wilensky et al., 1984) utilized a formal language to represent knowledge, which requires the heavy cost of construction and maintenance and makes the scaling up quite difficult.

In contrast, along with the improvement of NLP, research activities which utilize natural language text as a knowledge base become popular, such as START (Katz, 1990), FAQ Finder (Cooper, 1996), and QA Track in TREC 8 (NIST and DARPA, 2000). These systems, however, basically produce one time response, and do not have a *conversation* with users.

This paper proposes a dialogue helpsystem in which natural language knowledge base is not only used for one time response, but also for conducting a conversation. To put it concretely, the system can

- ask the user back if there is an unknown word in the user utterance,
- interpret the user utterance contextually, and
- ask the user back when the user utterance is not clear.

In order to achieve these functions, a mild restriction is imposed on the knowledge base; it is compiled in a dictionary-like structure (entry and its description) and the description is written in an if-then format (though several expressions can be used to show if-then relation).

The dialogue helpsystem reported in this paper advises faculties and students in using the computer facilities and software applications of a university. The computer facilities of Kyoto University, about 1,200 computers (PCs and WSs) in several rooms and modem connections are administrated by the Center for Information and Multimedia Studies (CIMS; http://www.media.kyoto-u.ac.jp/). The situation at CIMS is that many people are using computers, teaching

assistants (TAs) help the users, but are not always available, and certain fundamental questions are asked frequently. Considering this, we assumed that it is a very promising domain for an experimental dialogue system.

The system started its service on July 1999, received about 2,000 queries for the first seven months, and answered about 40% of them satisfactorily.

2 Examples

The following examples (real logs of the service) are illustrative of the breadth of dialogues in which the system aims to participate. The system is capable of generating the answers shown below in response to most forms of the request users might supply[1].

<1999 Jul 08 19:31:49>
U1: Method to invoke a screen save in UNIX?
S1: What is "screen save"?
U2: screen saver.
S2: Type "xlock" in the command line.

<1999 Aug 19 21:20:52>
U1: Please tell me how can I read emails in Emacs.
S1: Select mail software from "mh-e", "mew".
U2: I try mh-e.
S2: Type "M-x mh-rmail" on Emacs. Type "i" to get unread emails and type "." after you move the cursor on the line you want to read.

<1999 Sep 16 22:32:20>
U1: How can I use Mew in UNIX?
S1: After invoking Emacs, type "M-x mew". See http://www.mew.org/ for details.
U2: How can I delete emails?
S2: Type "d" in the Summary Mode and then type "x" for execution.

[1]Although our system is a Japanese dialogue system, in this paper we use their English translations for the explanation.

<1999 Oct 25 12:42:28>
U1: Tell me how to use Netscape.
S1: On the [Start Menu], click [program] and then click [Netscape Navigator].
U2: What's start menu?
S2: [Start Menu] is a pull-down menu summarizing Windows applications, which appears when you click [Start] icon on the left bottom of the desktop screen.
U3: What is click?
S3: I cannot explain "click" in detail.

3 Types of user utterances

We observed the conversations of users and TAs in the CIMS computer rooms by recording and transcription (20 hours observation; 1.5 hours recording).

From this study, we discovered that user utterances can be classified as follows:

What type: The user asks some fact.

ex. What is Emacs?
Explain Emacs.

How type: The user asks how to do something.

ex. How can I input Japanese characters in Emacs?
I want to input Japanese characters in Emacs.

Symptom type: The user shows some symptom since he/she want to know how to cope with it.

ex. I cannot access my private folder.
The screen is white.

Request type: The user requests something to CIMS.

ex. Please install Visual-C.
Please make my disk quota bigger.

Addition type: The user adds or modifies the previous question.

ex. How about WindowsNT?
In the case of reply?

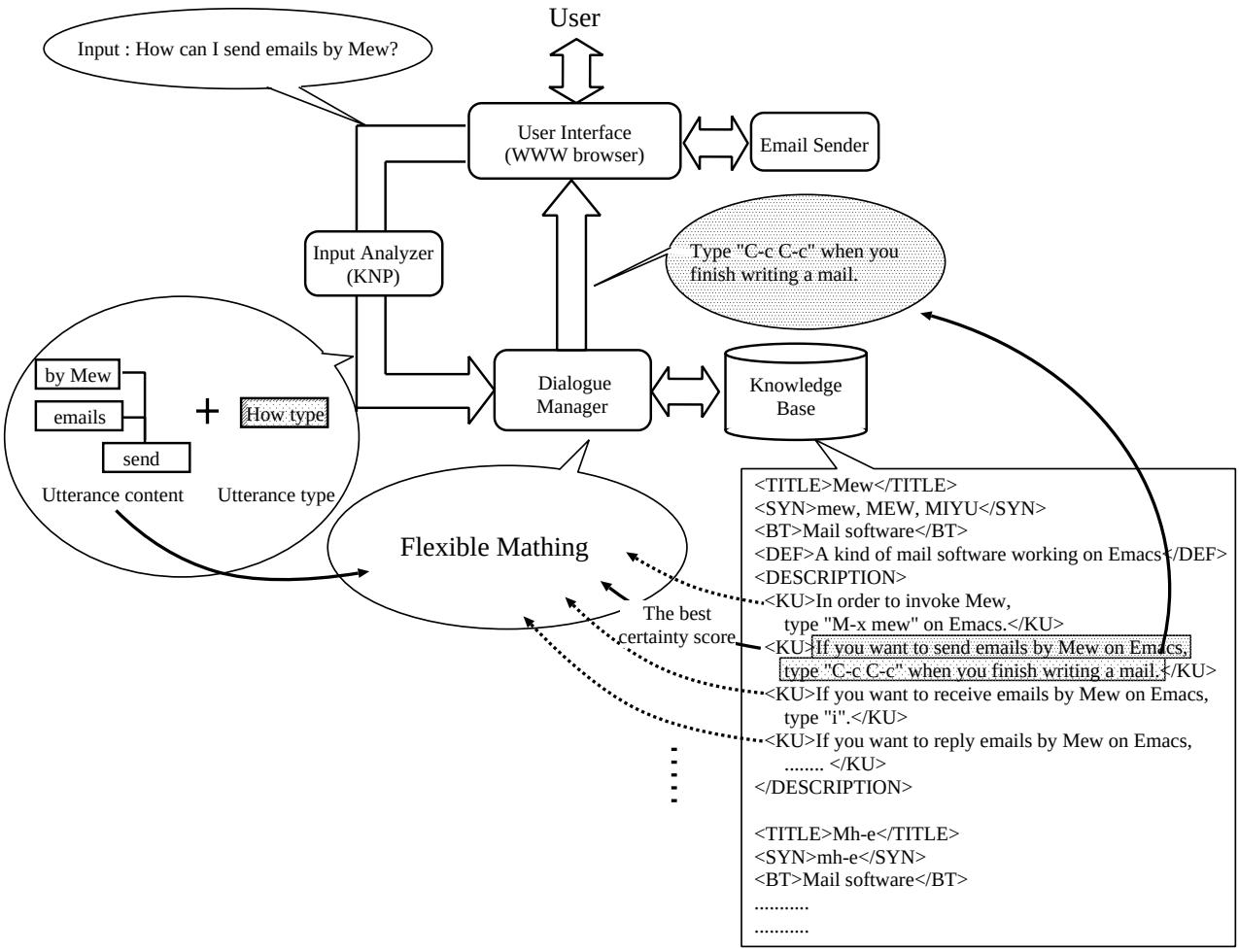


Figure 1: The outline of the helpsystem.

Answer type: The user answers the system question.

ex. WINDOWS.
The compression type is zip.

The helpsystem reported in the paper answers *what*, *how* and *symptom* questions. In addition, it can interpret *addition* and *answer* utterances contextually. The *request* utterances are out of the system scope currently.

4 Outline of the helpsystem

The system is comprised of the following components (Figure 1):

User Interface: Users access to the helpsystem via a WWW browser by using CGI based HTML forms. The helpsystem is actually running on a workstation in our lab.

Input Analyzer: The user utterance is transformed into a dependency structure by a robust parser, KNP (Kurohashi and Nagao, 1994; Kurohashi and Nagao, 1998), and utterance-pattern rules are applied to extract the utterance type and the utterance content.

Japanese is head-final and the final expression shows an utterance type. Therefore, the longest matching of utterance-pattern rules form the end of the utterance can detect the utterance type in most cases. For example, if the final expression is "*niha doushitara ii desu ka* (How can I $\cdots$)", *how* type is assigned; if "*no baai ha* (In case $\cdots$)", *addition* type.

Knowledge base: The knowledge base is written in a natural language, in a dictionary-like structure.

Dialogue Manager: The core process of the dialogue manager is to match the user utterance with the knowledge base in order to find the most appropriate description. It also handles contextual interpretation of the user utterance and question to the user.

Email Sender: The user can send his/her input-log to the CIMS staff via email if the automatic response is not satisfactory. So, the user does not have to input his questions a second time. This option surely contributes to the popularity of the system.

In the following sections, we discuss the knowledge base and the dialogue manager, since we consider these components as the core of the system.

5 Knowledge base

5.1 The outline

The knowledge base has a dictionary-like structure, in which each entry describes a concept/issue in the domain. It was compiled manually by referring to the real QAs in CIMS rooms, the FAQ page of CIMS (about 100 items), and question emails sent to CIMS (about 150 emails). Currently, it contains about 250 entries.

Each entry consists of a headword (<TITLE> tag), synonyms(<SYN> tag), an upper word (<BT> tag), a definition of the headword (<DEF> tag) and several descriptions concerning the headword (<DESCRIPTION> tag; see Figure 1). All content words in the knowledge base were registered to the system database, which is used to see whether a user input word is known or unknown to the system (Section 6.1).

In addition, by collecting the headword and its upper word pairs from the knowledge base, the domain ontology (concept taxonomy) is constructed automatically. The top categories of the current ontology are *software*, *hardware*, *computer term* (different to *soft/hardware*), *action term*, and *general term*. The domain ontology is used by the dialogue manager in several ways (Section 6.2, 6.3).

5.2 Natural language representation

In the knowledge base, the definition and several descriptions for the headword are written in natural language, which provides both high power of expression and high extensibility.

The definition of the headword is used for *what* questions; the descriptions are used for *how* and *symptom* questions. Each description, called *knowledge unit* (abbreviated to KU), is written in the following style:

<KU> if *a case*, then *what/how to do.* </KU>

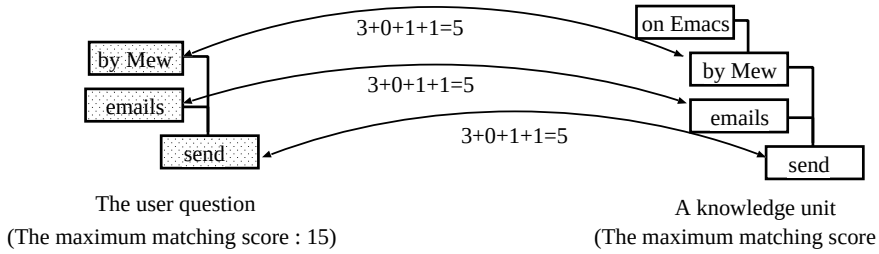
In Japanese, there are many sentential patterns to express if-then relation. Authors of the knowledge base can use several expressions like "$\cdots$ *deareba* $\cdots$ (if $\cdots$, $\cdots$)", "$\cdots$ *no baai ha* $\cdots$ (in case that $\cdots$, $\cdots$)".

The basic form of the *how* and *symptom* question is "in some case, what/how can I do?". Therefore, the system can answer the question by finding the most similar KU *case* part and showing the corresponding *what/how to do* part to the user (see Figure 1).

5.3 Matching of user question and knowledge unit

Matching of the user question and a knowledge unit (KU) is done by comparing their dependency trees whose nodes are phrases. Their similarity is calculated as follows (Figure 2):

1. For each phrase in the user question, the most similar phrase in the KU *case* part is looked for based on the following criteria:
   - Matching of content words : 3 points
   - The second or more matching of content words (when the phrase contains two or more content words) : 1 point



$$\text{The certainty score} = \frac{(5+5+5)^2}{15 \times 20} \times 100 = 75\ (\%)$$

Figure 2: Matching of the user question and a knowledge unit.

- Matching of the depth of the phrases in parse trees : 1 point
- Matching of the type of the phrases (phrase types differ depending on surface cases and verb conjugations, etc) : 1 point

2. The similarity scores of phrases in the user question are summed up and normalized by the maximum matching score (MMS) as follows (the MMS is the similarity score with the same sentence):

$$\frac{\left(\begin{array}{c}\text{The sum of scores of}\\ \text{phrase similarities}\end{array}\right)^2}{\left(\begin{array}{c}\text{The MMS of}\\ \text{the user question}\end{array}\right) \times \left(\begin{array}{c}\text{The MMS of}\\ \text{the KU } \textit{case}\end{array}\right)}$$

The above score is given to the KU as its certainty score.

The above algorithm cares for the structures of sentences to some extent by giving phrase depth scores and phrase type scores, but not in a strict way. This leads to a flexible matching for handling a variety of natural language sentences and some parse errors. For the present, the parameters were given empirically.

6 Dialogue manager

Figure 1 showed the simplest case of a QA. In some cases, however, the user and the system have to take more turns until the user obtains a satisfactory answer. Such a turn-taking is an essential point of a conversation.

To conduct a conversation, that is, to perform a proper turn-taking, the participant have to be able to do the following functions at least:

- ask the opponent back if there is an unknown word in the opponent's utterance,
- interpret the opponent's utterance contextually, and
- ask the opponent back when the opponent's utterance is not clear.

Our dialogue helpsystem can perform the basic level of the above functions by referring to natural language knowledge base. In the following subsections, we explain each of these functions in detail.

6.1 Asking back of an unknown word

Given the user utterance, the system first checks whether each content word in it is registered in the system database or not. If the word is not registered, it means that the word is an unknown word to the system. An unknown word appears in the following cases:

1. Technical term not covered by the knowledge base.
   ex. shell script, clone
2. Technical term whose synonym or related term is covered by the knowledge base.
   ex. Mozaic, Mozilla
3. Misspell of the user.
   ex. Internetmai, Windo

Table 1: Patterns of the system responses.

The best certainty score	# of the candidate KUs	
	one	many
100–60%	<what/how to do> (of the KU)	(one difference) Select <upper concept> from <list of the difference>. (two or more differences) Your question is not clear. Select <list of the candidate KU cases>.
60–30%		Your question is not clear. Select <list of the candidate KU cases>
30–0%	I cannot answer your question.	

4. General term.
   ex. name, summer vacation

The system decision, whether the unknown word is general term or not, is taken according to whether it is an entry of a children's dictionary or not (Tadika, 1997).

If the unknown word is not a general term, the system asks the user back in the form "what is 'unknown word'?". If the system asks "what is Internetmai?", the user probably notices his/her misspell and re-input it correctly. If the system asks "what is Mozilla?", the user might paraphrase it like "It means Netscape".

If the unknown word is a general word, it does not make sense to ask the user back, like "what is name?". Therefore, the system just overlooks it.

6.2 Contextual interpretation of user questions

The user question may be related to its preceding utterances, modifying or supplementing them. As an example, consider the following dialogue:

U1: How can I send emails by Mew?
S1: Type "C-c C-c" when you finish writing a mail.
U2: In case to reply.
S2: Move the cursor on the message to which you want to reply and type "A".

In this dialogue, the user utterance U2 is a modification of U1, indicating "How can I reply emails by Mew?".

In order to interpret such a context dependent utterance properly, the dialogue manager attempt to merge the user's new utterance onto the previous one. For each word of the user's new utterance, w_{new}, if the previous utterance contains the word of the same category, w_{old}, w_{new} is overwritten on w_{old}. If not, w_{new} is added to the previous utterance. Two words are considered to be in the same category if they belong to the same top category of the domain ontology described in Section 4.1. Then, the system looks up the knowledge base by the merged utterance.

In the above example, "reply" of U2 is overwritten on "send" of U1, since they belong to the same category, *action term*. Then the system attempts to match the combined utterance "How can I reply emails by Mew?" with the knowledge base, and as a final result, it can response as S2 [2].

In the above example, since U2 is an *addition* utterance, the system does not need to interpret U2 as a new, context independent question. However, if the user utterance has a different type, it is not possible to decide whether it is context dependent or indepen-

[2]Note that the system keeps the resultant interpretation of the user query, which means that the system can keep more than one user utterances practically. For example, if the user asks "In case to forward" after S2 in the above example, the system can interpret it as "How can I forward emails by Mew".

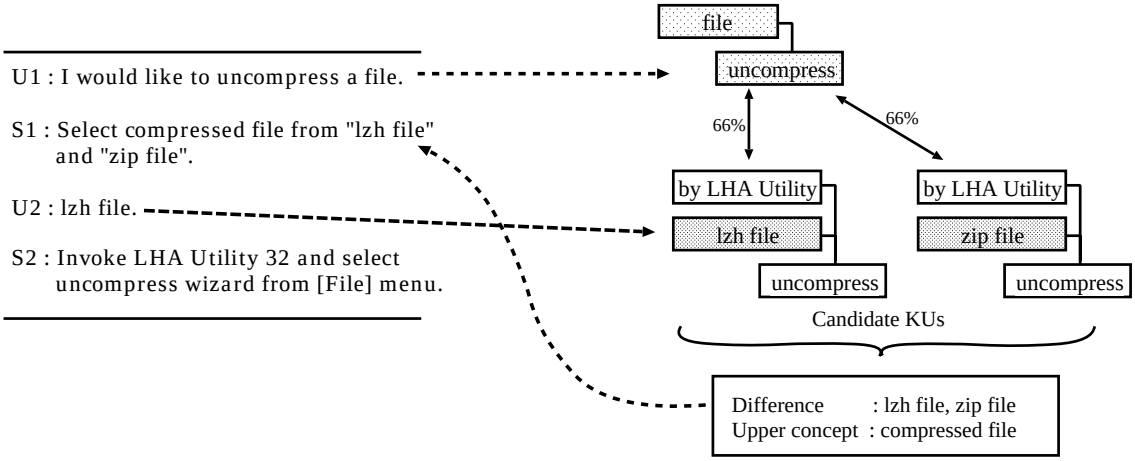


Figure 3: An example of handling many candidate KUs.

dent by seeing the new utterance alone.

To decide this, the dialogue manager regards the certainty score between the utterance and the most similar KU as an appropriateness measure of the interpretation. That is, we calculate both the certainty score between the merged utterance (contextual interpretation) and the knowledge base, and that between the new utterance as it is and the knowledge base. Then, we choose the interpretation which obtained the KU with the bigger certainty score.

For example, in the following dialogue, U2 is interpreted as "How to invoke Mew?", since this combined interpretation has the bigger certainty score rather than U2 itself.

U1: What is Mew?
S1: A kind of mail software working on Emacs.
U2: How to invoke?
S2: Type 'M-x mew" on Emacs.

On the other hand, in the next example, U2 is interpreted as it is, since U2 has the bigger certainty score rather than the combined, somehow strange interpretation of "How to uncompress zip files by Outlook on Unix?".

U1: I want to send an email by Outlook.
S1: Invoke OutlookExpress, write your email, and select "Send/Get" button.
U2: How to uncompress zip files on UNIX.
S2: Type "unzip [zip filename]" in the command line.

6.3 Asking back for an unclear question

As mentioned so far, the system retrieves proper KUs which are similar to the user question. The KU with the best certainty score and the KUs with the 90% or larger certainty score are called *candidate KUs*. The dialogue manager performs differently depending on the best certainty score and the number of candidate KUs, as shown in Table 1.

If the certainty score is 60% or higher and the number of the candidate KUs are two or more, the dialogue manager detects the difference between their cases and ask the user to make his/her question more specific.

Figure 3 shows an example of such a dialogue. The two candidate KUs are detected for U1 and their cases have only one difference: "lzh file" and "zip file". Then, the system detects their common upper concept in the domain ontology and ask the user in the form "Select <upper concept> from <list of the difference >" as shown in Figure 3.

If the candidate KUs contain two or more differences, it is hard to edit them in a neat way. Therefore, the system shows the candidate KUs' cases as they are. If the certainty score is less than 60% and larger than 30%, the system responses in the same way.

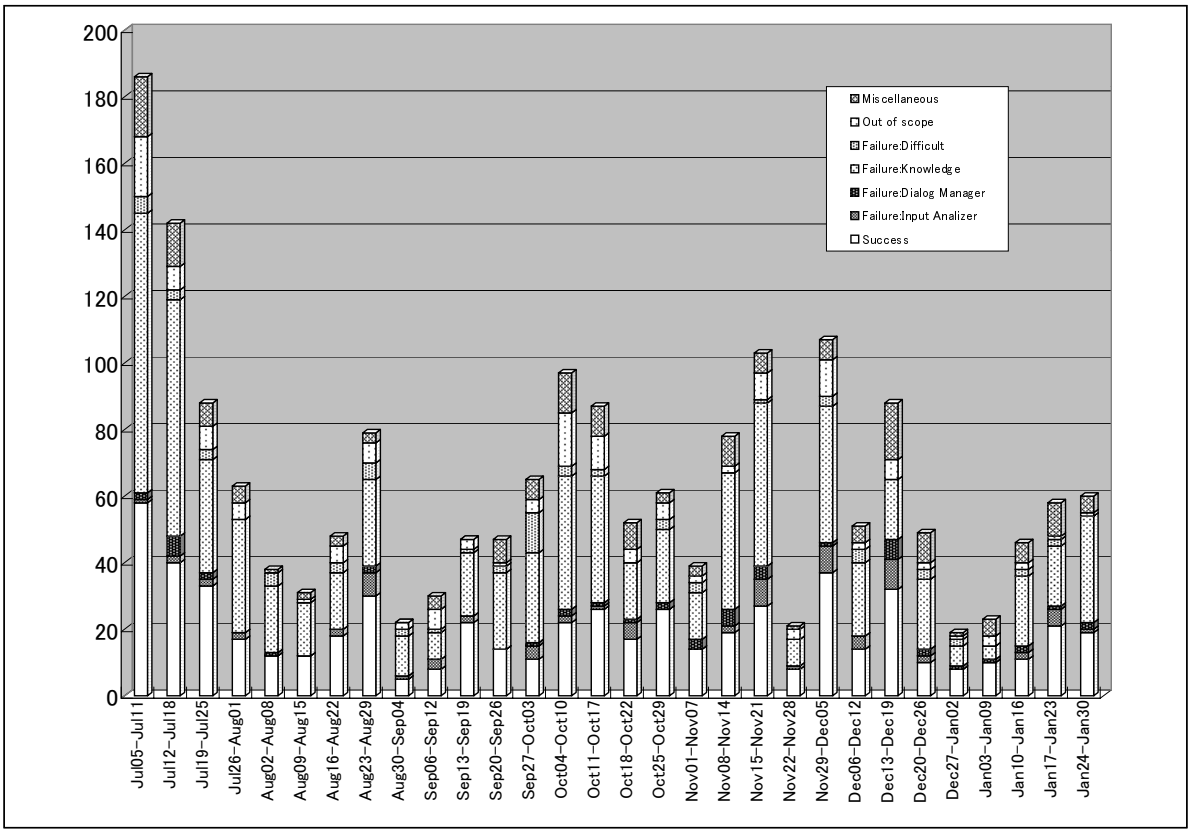


Figure 4: Evaluation of the helpsystem.

7 Evaluation

The helpsystem started its service on July 1999 as part of the CIMS web service. All conversation logs between users and the system have been stored as a dialogue database.

In the dialogue database, each dialogue between a user and the system is segmented into task units manually. We call this unit a *session*. Figure 4 shows the number of sessions and their evaluation of each week from July 5th to January 30th. On average, there are 70 sessions in a week; a dialogue with a user and the system consists of 2.1 sessions, which means a user asks 2.1 topics in one dialogue; one session consists of 3.2 turns.

The evaluation of sessions is based on the following criteria.

Success: The system could return a satisfactory answer.

Failure:Input Analyzer: The system could not response properly because of the input analysis error, mostly the lack of utterance-pattern rules. Utterance-pattern rules are added whenever the lack is found.

Failure:Dialog Manager: The system could not response properly because of the dialogue manager error. Dialogue manager error comes both from simple bugs of the system and from unnoticed patterns of the user response. For example, when the system asks "select from A and B" expecting the answer "A" or "B", a user might answer "the latter". The system is modified whenever necessary.

Failure:Knowledge: The system could not answer the question because of the lack of knowledge. This is the major reason of the failure as shown in Figure 4. Though the knowledge base is being extended step by step, the range of the user

query is unlimited, including troubles in using PCs and advanced settings of software/hardware.

Failure:Difficult: Current system architecture could not handle the question. For example, a user sometimes asks "what is the difference between A and B", or when the system asks "select from A and B", a user answers "I don't know". In order to handle such utterances, we are planning to improve the system to exploit definitions of "A" and "B".

Out of scope: Out of the system domain, such as questions about telephone charges in using PPP or the Y2K problem.

Miscellaneous: Such as "hello", "this is a test" or just a simple typo like "a".

The success ratio, that is, the ratio of Success over Success plus Failure, of the whole period is 37%. The system became stable around October 1999, and the success ratio after that (14 weeks) is 39%. Considering relatively wide domain the system have to cover, we feel the success ratio is reasonable, and the system is contributing to CIMS to some extent by handling simple FAQs like "how to change my password".

8 Conclusion

This paper described the dialogue helpsystem, which has been working in practice with real users.

Construction of natural language knowledge base needs some cost, though it is much easier than that of formal language knowledge base. However, providing a high-quality service needs cost; good manuals and FAQs are important for any products, and a large amount of materials are prepared for customer service operators. With that in mind, we can say preparing a good document is a universal problem, not just to a dialogue system.

By running the system, the real dialogue database can be accumulated. Based on this database, we would like to study the phenomena of man-machine conversation and to extend our work to user modeling, user intention estimation, and other interesting dialogue research areas.

The system is designed to be domain-independent and can be ported to a new domain by preparing a domain knowledge base. Exploiting this merit, we are planning to construct the automatic reference service system of Kyoto University Library, which certainly provides us with a wider breadth of dialogue data.

References

Edwin Cooper. 1996. Improving FAQ Finder's performance: Setting parameters by getetic programming. In *Working Notes of the AAAI Spring Symposium on Machine Learning in Information Access*.

B Katz. 1990. Using english for indexing and retrieving. In *Artificial Intelligence at MIT. Vol.1, MIT Press*, pages 134–165.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4).

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proceedings of the First International Conference on Language Resources & Evaluation*, pages 719–724.

NIST and DARPA. 2000. *The Eighth Text REtrieval Conference (TREC-8)*. NIST Special Publication.

Jyunichi Tadika, editor. 1997. *Reika Shougaku Kokugojiten (Japanese dictionary for children)*. Sanseido.

Robert Wilensky, Yigal Arens, and David Chin. 1984. Talking to unix in English: An oveview of UC. *Communications of the ACM*, 27(6):574–593.

Yorick Wilks, editor. 1999. *Machine Conversations*. Kluwer Academic Publishers.