Preliminary Evaluation of Speech-to-Text Query Application for Parts Database

Yuichiro Aoki
Research and Development Group, Center for Technology Innovation - Digital Technology
Hitachi, Ltd.
Tokyo, Japan
email: yuichiro.aoki.jk@hitachi.com

Tadashi Takeuchi
Research and Development Group, Data Science Laboratory
Hitachi, Ltd.
Tokyo, Japan
email: tadashi.takeuchi.dt@hitachi.com

***Abstract*— To ensure the safety and security of workplaces and homes, parts replacement of mechanical/electric/electronic devices is inevitable, because they will gradually break down. Users have to search for the appropriate part numbers from a parts database to order and replace the parts. In many cases, users make phone calls to the support center of the vendors instead of searching by themselves, which can be inconvenient for the support center staff. In this paper, to reduce the number of phone calls to support centers, we propose a prototype of an automatic part number answering system from natural language. This system consists of open-source speech-to-text conversion and Structured Query Language (SQL) generation from the natural language. Preliminary evaluation results show that 83% of the voice questions returned the correct part numbers. In addition, the search with our system was executed an average of 3.86 times faster than with the conventional manual keyword search.**

Keywords-speech-to-text; SQL generation; natural language processing.

I. INTRODUCTION

To ensure the safety and security of workplaces and homes, parts replacement of mechanical, electric, and/or electronic devices in factories, construction sites, and homes is inevitable, as such devices will gradually deteriorate and break down. Before a malfunction occurs, device users have to search for the part numbers on a parts database to order and replace parts. However, generally speaking, users do not know the correct part names, thus they cannot use the conventional manual keyword search. Instead, they make phone calls to the support center of the device vendors, which inconveniences the support center staff, who invariably have little time to improve their productivity or the quality of their support. To reduce the number of phone calls or even completely eliminate them, one potential solution is an automatic answering system for the part number.

In this paper, we propose an automatic part number answering system using speech recognition (speech-to-text) and SQL generation from natural language and make a preliminary evaluation of its functionalities and performance. We use an open-source software for speech-to-text and develop a SQL generation algorithm. More specifically, we assume that "ask back" processing of the system is new and more pragmatic than previous studies in real industry fields.

In Section II of this paper, we review related work. In Section III, we describe the architecture of our system and Graphical User Interface (GUI). The preliminary evaluation results are reported in Section IV. In Section V, we discuss the functionalities and the performance, followed by a conclusion and future study in Section VI.

II. RELATED WORK

In this section, we review speech-to-text technology and SQL generation technology from natural language. First, we focus on speech-to-text. Table I shows the comparison of speech-to-text technologies offered by various providers. Amazon Transcribe on Amazon Web Services (AWS) handles 14 languages, but does not handle Japanese and has no sound model customization [1]. Cloud Speech-to-Text on Google Cloud Platform (GCP) can recognize 120 languages and dialects, including Japanese, and can customize the sound model. However, it does not have a user dictionary [2]. Watson Speech to Text on IBM Cloud handles Japanese, has a user dictionary, and can customize the sound model, but is proprietary [3]. Speech-to-Text on Microsoft Azure handles Japanese and can perform sound model customization. However, it does not have a user dictionary [4].

TABLE I. COMPARISON OF SPEECH-TO-TEXT TECHNOLOGIES.

Name	*Japan-ese*	*User Dictionary*	*Sound Model Custom-ization*	*Open Source Software*
Amazon Transcribe		✓		
Google Cloud Speech-to-Text	✓		✓	
IBM Watson Speech to Text	✓	✓	✓	
Microsoft Speech-to-Text	✓		✓	
Julius (This Study)	✓	✓	✓	✓

 ISBN: 978-1-61208-798-6

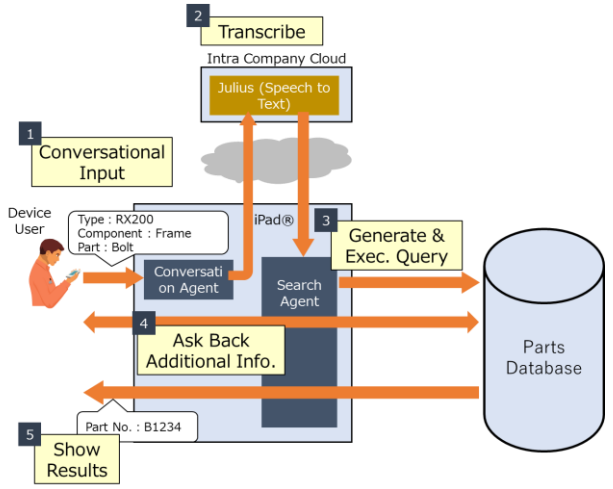


Figure 1. Overview of our system.

Julius is an open-source, speech recognition engine with a large vocabulary. It supports statistical N-gram model and rule-based grammars as a language model and Hidden Markov Model as a sound model [17]. Julius translates Japanese speech into Japanese text, has a user dictionary, can customize the sound model, and is open-sourced [5]—in fact, no speech-to-text technology other than Julius is open-sourced. In addition, Julius can be deployed on the intra-company cloud and has less risk of information leak. For these reasons, we opt to utilize Julius for our system.

Second, we examine SQL generation technologies from natural language [6][7]. Precise [8] constructs systematic natural language queries that are easy to translate into SQL queries. Rao et al. [9] convert inquiries that are written in natural language and whose words are in the user-defined dictionary into SQL queries. They cannot make queries interactively. Safari and Patrick [10] transform general natural language queries into restricted natural language queries first and then convert them into SQL queries. NaLIR [11] makes a syntax analysis of the natural language query and shows its result to the user. Next, the user manipulates the result manually and modifies the syntax tree to one that is easy to convert into an SQL query. Sukthankar et al. [12] generate complex SQL queries that include WHERE clauses. Seq2SQL [13] converts natural language queries into SQL queries by using reinforcement learning. However, the accuracy of the generated SQL queries is about 50–60%. All the systems above lack interactive functionalities on mobile devices or have insufficient accuracy of the generated SQL queries. In contrast, our system interactively converts the natural language questions into SQL queries on mobile devices such as an iPad® and does not show the details of the analysis or the transformation. Moreover, it can generate complex SQL queries using WHERE clauses. Preliminary evaluation results show that 83% of the generated SQL queries are correct.

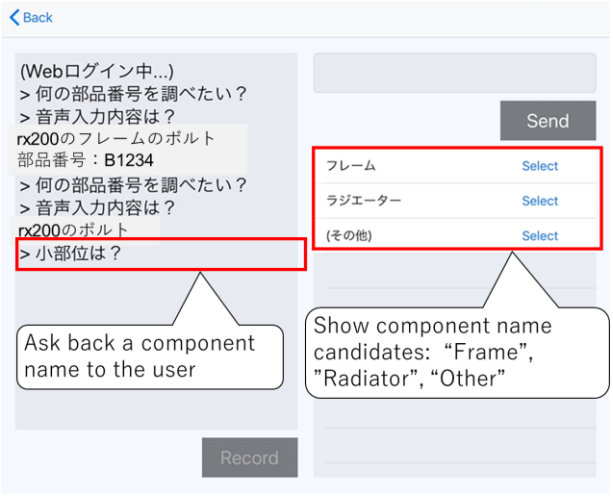
Real world applications, such as Apple Siri [14], Google Assistant [15], and Amazon Alexa [16], resemble our system. However, they cannot use intra-company proprietary databases. Our system can deal with such databases.

III. Overview of Our System

In this section, we provide an overview of our system. The device user makes a conversational input consisting of the part information, such as the machine type name (RX200), the component name (Frame), and the part name (Bolt), to the system. The system analyzes the information and returns the corresponding part number (B1234).

This process is shown in Figure 1. First, in step 1, the user talks to the conversation agent on the iPad®, saying (for example) “RX200, Frame, and Bolt”. In step 2, the conversational agent accepts the utterance, formats it as Moving Picture Experts Group (MPEG)-1 Audio Layer-3 (MP3) and sends it to the speech-to-text engine (Julius) on the intra-company cloud.

(a) Complete question



(b) Incomplete question

Figure 2. GUI images on an iPad®.

Julius transcribes the utterance to text and sends it back to the search agent on the iPad®. In step 3, the search agent generates the SQL query. Words from the text are embedded in WHERE clause. Finally, the search agent executes the query on the parts database. The example SQL is as follows:

SELECT part_number FROM parts_database WHERE type='RX200' AND component='Frame' AND part='Bolt'.

The parts database returns the part numbers to the search agent. If the search agent does not get a unique part number from the SQL query, in step 4, it generates an additional SQL query from the answer, as follows:

SELECT component FROM parts_database WHERE type='RX200' AND part='Bolt'.

The answer of the additional SQL query is displayed as selection buttons on the iPad®. The device user taps the appropriate button for the component name, that is, the search agent asks back the device user what the component name is. Finally, the search agent shows the part number, which in this case is "B1234", on the iPad® in step 5. Manual correction of the speech recognition result can be performed if the user finds incorrect results.

Next, we present the GUI of our system. Figure 2(a) shows an example where a complete question is asked, which includes a type name, a component name, and a part name. The user verbally inputs the complete question and the system transcribes it (red box in the figure). Then, the user selects the appropriate text and the system replies with the part number.

The case of an incomplete question is shown in Figure 2(b), where some of the required information is lacking in the question—in this example, a component name. The system analyzes this incomplete question and shows candidates for necessary information that is missing in the voice input ("ask back" processing, red boxes in the figure). The user taps the appropriate candidate button and the system replies with the part number.

IV. PRELIMINARY EVALUATION

In this section, we report our preliminary evaluations of the functionalities and performance of our system. All evaluations are done in Japanese. We use the Apple® iPad® (6th Generation) and the intra-company cloud for the evaluation.

A. *Functional Evaluation*

We manually make 100 artificial oral questions to test the coverage of the system. Each question is composed of a machine type name, a component name, and a part name. That is, all questions are complete question cases. All combinations of the names in the questions are different from each other. Examples of questions are "RX200, Frame, and Bolt" or "RX78, Generator, and Chain". One of the authors orally inputs these questions in the office.

Table II lists the functional evaluation results. As shown, the system delivered the correct part numbers for 83 of the questions. However, 12 questions exhibited inappropriate speech recognition, and there were five error results.

TABLE II. FUNCTIONAL EVALUATION RESULTS.

Results	Number of Questions
Correct part number	83
Inappropriate speech recognition	12
Other errors	5
Total	100

TABLE III. BREAKDOWN OF 83 SUCCESSFUL RESULTS

Voice Input	Additional Info Needed	Manual Correction	Number of Questions
✓			0
✓	✓		2
✓		✓	40
✓	✓	✓	41

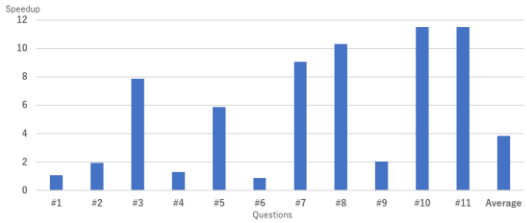


Figure 3. Speedup over the conventional search.

Table III shows the breakdown of the successful 83 results. In the case of voice input only, there were no results. When we used the voice input and the additional information input required by the system ("ask back" processing), there were two results. Using the voice input and the manual correction of the small speech recognition mistake, there were 40 results. When all three items were used, there were 41 results.

B. *Performance Evaluation*

Next, we evaluate the performance of the system. We prepare 11 additional oral questions from the device users. These 11 questions are different from the 100 questions in the previous section in that they were actually used by the device users in real engineering fields. Some questions are incomplete, such as "RX78 and Generator". One of the authors orally inputs these questions in the office, too. Figure 3 shows the speedup of the search time over the conventional search. The conventional search specifies the correct part name manually using a keyboard and then searches for it. If the search does not get the part number within 300 seconds, we assume the search time to be 300 seconds. When we tallied the results, we found that our system had a speedup larger than 1 in all questions except #6 (0.88 times). The average speedup was 3.86.

 ISBN: 978-1-61208-798-6

V. DISCUSSION

In this section we will discuss the functional and performance evaluation results.

A. Functional Evaluation Results

First, our system could not transcribe 12 questions properly due to a deficiency of the synonym dictionary for terms such as “urea”. We need to expand the synonym dictionary to include domain-specific terms. The other five errors are currently under investigation.

Second, 81 of the 83 questions that led to the correct part number required manual correction of the transcribed text. This is because Julius failed to detect the pauses between words, as the background noise of the evaluation environment was not exactly small. A noise-resistant system is a future consideration. Parameter tuning of Julius may improve the situation.

Third, 43 questions required additional information to reach the correct part number. This is because the same part and component names appeared in a device type. In such cases, the system asks back the device user which information is needed. To automate this issue, we propose to display probable combinations of names and the user to select proper one.

B. Performance Evaluation Results

In the conventional search, the device user needed to know the correct type/component/part name before the search. In addition, some questions lacked information (e.g., component name). As a result, the user could not input the correct search keywords and six of the 11 questions did not lead to correct part numbers within 300 seconds. In contrast, our system could obtain all the correct part numbers within 300 seconds. In several cases, questions about the missing information were automatically asked back by the system. In question #6, the tester coincidentally knew the correct part name, so the conventional search was executed faster than our system.

Additionally, our system acquired the part numbers an average of 3.86 times faster than the conventional search. We assume that the automatic “ask back” processing in our system helped reduce the search time.

VI. CONCLUSION

To reduce the number of phone calls to support centers, we have developed an automatic part number answering system consisting of an open-source speech-to-text software and SQL generation from natural language (Japanese).

The functional evaluation results demonstrated that 83% of the questions returned the correct part numbers. Moreover, the performance evaluation results showed that our system executed the search an average of 3.86 times faster than the conventional search.

In future work, we will examine how to improve the accuracy of voice recognition under noisy environments and decrease latencies of the system for speedup. In addition, we are planning to evaluate the performance and robustness of the SQL generation algorithms.

ACKNOWLEDGMENT

The authors thank Dr. Tsuyoshi Tanaka and Dr. Hiroaki Shikano for supporting this research.

REFERENCES

[1] Amazon Web Services, “Amazon Transcribe” [Online]. Available from: https://aws.amazon.com/transcribe/ 2020.03.26

[2] Google Cloud, “Cloud Speech-to-Text” [Online]. Available from: https://cloud.google.com/speech-to-text 2020.03.26

[3] IBM, “Watson Speech to Text” [Online]. Available from: https://www.ibm.com/cloud/watson-speech-to-text 2020.03.26

[4] Microsoft, “Speech-to-Text” [Online]. Available from: https://azure.microsoft.com/en-au/services/cognitive-services/speech-to-text/ 2020.03.26

[5] Julius, “Open-Source Large Vocabulary Continuous Speech Recognition Engine” [Online]. Available from: https://github.com/julius-speech/julius 2020.03.26

[6] B. Sujatha, S. V. Raju, and H. Shaziya, “A survey of natural language interface to database management system,” International Journal of Science and Advanced Technology, vol. 2, no. 6, pp. 56–61, 2012.

[7] H. S. Dar, M. I. Lali, M. U. Din, K. M. Malik, and S. A. C. Bukhari, “Frameworks for Querying Databases Using Natural Language: A Literature Review,” arXiv:1909.01822, 2019.

[8] A.-M. Popescu, O. Etzioni, and H. Kautz, “Towards a theory of natural language interfaces to databases,” in Proc. of the 8th International Conference on Intelligent User Interfaces, pp. 149–157, 2003.

[9] G. Rao, C. Agarwal, S. Chaudhry, N. Kulkarni, and D. S. H. Patil, “Natural language query processing using semantic grammar,” International Journal of Computer Science and Engineering, vol. 2, no. 2, pp. 219–223, 2010.

[10] L. Safari and J. D. Patrick, “Restricted natural language based querying of clinical databases,” Journal of Biomedical Informatics, vol. 52, pp. 338–353, 2014.

[11] F. Li and H. V. Jagadish, “Constructing an interactive natural language interface for relational databases,” in Proc. of the VLDB Endowment, vol. 8, no. 1, pp. 73–84, 2014.

[12] N. Sukthankar, S. Maharnawar, P. Deshmukh, Y. Haribhakta, and V. Kamble, “nQuery: A Natural Language Statement to SQL Query Generator,” in Proc. of ACL2017, Student Research Workshop, pp. 17–23, 2017.

[13] V. Zhong, C. Xiong, and R. Socher, “Seq2SQL: Generating Structured Querues from Natural Language using Reinforcement Learning,” arXiv:1709.00103, 2017.

[14] Apple, “Siri” [Online]. Available from: https://www.apple.com/siri/ 2020.08.14

[15] Google, “Google Assistant” [Online]. Available from: https://assistant.google.com/ 2020.08.14

[16] Amazon, “Alexa” [Online]. Available from: https://developer.amazon.com/en-US/alexa 2020.08.14

[17] A. Lee and T. Kawahara, “Recent Development of Open-Source Speech Recognition Engine Julius,” Asia-Pacific Signal Information Processing Association Annual Summit and Conference, 2009.

 ISBN: 978-1-61208-798-6