EMPLOYING LARGE LANGUAGE MODELS FOR DIALOGUE-BASED PERSONALIZED NEEDS EXTRACTION IN SMART SERVICES

Takuya Nakata[1], Sinan Chen[2], Sachio Saiki[3] and Masahide Nakamura[2]

[1]Graduate School of Engineering, Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe, Japan
[2]Center of Mathematical and Data Sciences, Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe, Japan
[3]School of Data and Innovation, Kochi University of Technology, 185 Miyanokuchi, Tosayamada, Kami, Japan

ABSTRACT

Research concerning the personalization of services encompasses approaches such as machine learning and dialogue agents; however, the explainability of the recommendation process remains a challenge. Previous studies have proposed dialogue-based needs extraction systems utilizing the 6W1H need model, but extracting complex needs using simple natural language processing proved challenging. In this research, we embark on the development of an Application Programming Interface (API) that extracts user needs from natural language by leveraging the rapidly advancing Large Language Models (LLM), and on constructing a dialogue-based needs extraction system using this API. For evaluation, we conducted a verification on 100 needs with the aim of assessing the accuracy and comprehensiveness of the outputs from the needs extraction and restoration API. Through this study, it became feasible to extract needs with high accuracy and comprehensiveness from complex natural language using LLM.

KEYWORDS

Personalization, Need, Large Language Model, Natural Language Processing, Dialogue Agent

1. INTRODUCTION

With the widespread adoption of smart services in society, research on service personalization, which tailors service delivery and functionality to users' preferences, thereby creating added value, has become increasingly prevalent [1,2]. Two primary approaches exist in the study of individual adaptation: recommendations through machine learning and function recommendations and updates through dialogue [3]. Research has also been conducted on conversational recommendation systems, which integrate these approaches [4]. However, a challenge arises in explaining to users the rationale behind the recommendations [5,6].

Traditional research proposed the 6W1H need model, which expresses the user's need regarding how they wish to execute a service through seven elements: `where`, `when`, `who`, `whom`, `why`, `what`, and `how` [7]. This model aimed to address the challenge of explaining the recommendation process. In this framework, morphological analysis and syntactic analysis were used to extract 6W1H elements from user needs utterances. However, there remained issues in extracting complex needs and specific elements.

The aim of this research is to address the challenges identified in previous studies and to enhance the diversity and accuracy of needs extraction. A key idea revolves around utilizing the Generative Pre-trained Transformer (GPT) model of Large Language Models (LLM), which has seen an explosive increase in usage in recent years, for the construction of a needs extraction Application Programming Interface (API) and the reconstruction of the dialogue-based needs extraction system [8]. The approach of this research is as follows:

(A1) Construction of the needs extraction and restoration API using LLM.
(A2) Design of the overall architecture.
(A3) Design of the dialogue flow.

For evaluation, the proposed Needs Extraction API and Need Restoration API will be tested with 100 inputs to assess the accuracy and comprehensiveness of the outputs, and whether the 6W1H elements can be sufficiently extracted.

2. PRELIMINARIES

2.1. Personalization

In the digital domain, personalization refers to the process of modifying a system's functionality, interface, and content to enhance its relevance to an individual or a group of individuals. There are two primary approaches to achieve personalization: machine learning and dialogue. The machine learning approach involves deep learning based on a user's past service usage history, aiming to predict and recommend services the user may prefer [9,10]. However, this approach faces challenges in explainability since the recommendation process often operates as a black box. The dialogue-based approach updates the user model, which represents user preferences, through interactions between the user and a virtual agent [11]. Additionally, this approach provides service updates. Dialogue-based personalization has been implemented in devices such as smart speakers and healthcare agents [12]. In recent years, conversational recommendation systems have emerged as an approach that integrates both machine learning and dialogue. These systems allow for user inquiries and feedback through dialogue and recommend service functionalities based on dialogue logs, using machine learning.

2.2. Conventional Research: Interactive Needs Extraction System

In conversation-based recommendation systems, the explainability of the recommendation process, similar to personalized adaptation through machine learning, is a challenge. Previous research aimed to introduce an interpretable need model, situated between dialogue and machine learning, to make the system’s interpreted needs explainable to users. Specifically, they proposed a personal adaptation system as shown in Figure 1, with a particular focus on constructing a user needs extraction method. This proposed system extracts user needs through a dialogue-based user needs extraction method, which is then employed in machine learning to recommend services. In the extraction method from previous research, conversational content was gathered through voice dialogue agents, and needs data represented in a 6W1H format, which users could easily comprehend, were extracted and stored using natural language processing. The 6W1H need model consists of the following seven elements that depict how users want to utilize a service:

- `where`: The location where the service is performed.
- `when`: The time when the service is performed.

- `who`: The main entity performing the service.
- `whom`: The target of the service.

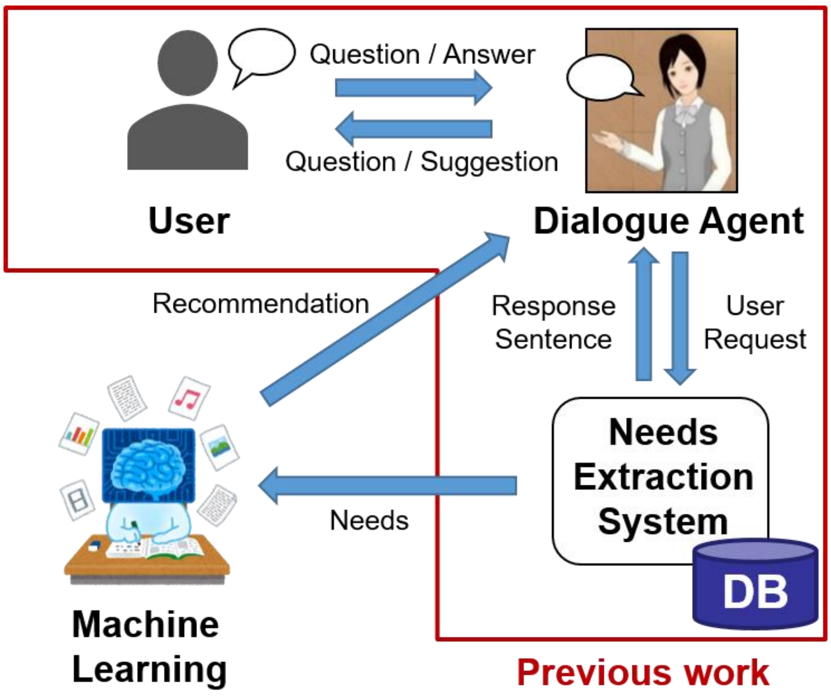

Figure 1. Overall architecture of the dialogue-based needs extraction system

- `why`: The reason or circumstances for performing the service.
- `what`: The specific action performed in the service.
- `how`: The method or tool used to execute the service.

This model employs the 6W1H analytical approach, a common framework for language usage, communication, and information organization, to address user needs effectively. Need data represented in the 6W1H format are referred to as 6W1H need data. For instance, the need "I want the smart speaker to notify my grandfather in the living room with a loud sound at 6:30 every morning if the weather forecast predicts rain" can be represented in 6W1H need data as:

- `where`: Living room.
- `when`: At 6:30 every morning.
- `who`: None.
- `whom`: My grandfather.
- `why`: If the weather forecast predicts rain.
- `what`: Notify with a loud sound.
- `how`: Smart speaker.

Traditional research used morphological and syntactical analysis to guess each element from the meanings of nouns and particles and to extract 6W1H need data. If some 6W1H elements were missing, the system would repeatedly ask the user questions like, "Where is the location?" to extract as detailed needs as possible from the user during the conversation. Finally, after extracting the needs, the system confirms with the user to ensure the extracted result matches the user's intended specifics.

2.3. Challenges in Previous Research

The needs extraction method using morphological and syntactical analysis in previous research has its limitations. Firstly, distinguishing between who and whom is challenging. Both elements represent people related to the execution of the service. However, determining whether one is the executor or the target of the service is difficult. Moreover, extracting the why element is problematic. It is challenging to discern the reasons or conditions for wanting to execute a service based solely on grammar. Additionally, extracting needs from complex sentences is demanding. It is not possible to accurately extract the 6W1H elements from utterances concerning multiple service executions. For instance, from the need expression, "I want to watch movies in the living room with a video service and do karaoke every morning with a music service," it is unclear, based solely on dependency relations, whether the where element pertaining to the living room is related to the video or music service.

2.4. Large Language Model

The LLM is a natural language processing model trained on extensive text data. It can be specialized for specific natural language processing tasks through fine-tuning [13]. GPT, developed by OpenAI, is one such LLM [14]. It is capable of various natural language processing tasks such as text generation, translation, and document summarization. In 2022, the Artificial Intelligence (AI) chat service ChatGPT was introduced, garnering significant global attention. ChatGPT is a service developed based on GPT-3.5 and GPT-4. With GPT, various processes can be executed by creatively designing natural language processing instructions using prompt engineering. The prominent method, Few-shot Prompting, enhances answer accuracy by providing several question and answer pairs as examples [15].

3. PROPOSED METHOD

3.1. Goal & Key Idea

The objective of this study is to enhance the diversity and accuracy of need extraction by extracting 6W1H need data from complex natural language expressions of needs communicated through interactions with virtual agents. The key idea revolves around the construction of a method that utilizes the LLM to extract 6W1H need data from intricate natural language expressions. The approach of this study is as follows:

(A1) Construction of the needs extraction and restoration API using LLM.
(A2) Design of the overall architecture.
(A3) Design of the dialogue flow.

3.2. (A1) Construction of the needs extraction and restoration API using LLM

We developed three distinct APIs utilizing the LLM to achieve the conversion between natural language and 6W1H need data: the Needs Extraction API, the Needs Re-extraction API, and the Need Restoration API. An API is a mechanism that allows programs to be invoked from outside the software. The implementation of these three types of APIs enables the extraction, re-extraction, and restoration of needs from various programs. The Needs Extraction API is designed to transform user needs expressed in natural language into 6W1H need data. The Needs Re-extraction API supplements missing 6W1H elements from previously extracted 6W1H need data using new user need utterances. The Need Restoration API reverts 6W1H need data back into a natural language expression of the need. For the API implementation, OpenAI's Chat API was employed. The model used is gpt-3.5-turbo, with a sampling temperature of 0.0. Few-shot Prompting was utilized for the prompt, providing several input-output examples. The programming language used for the API's development is Python.

```
[{
"where": "",
"when": "",
"who": "I",
"whom": "people who match my interests",
"why": "",
"what": "connect and share real-time content",
"how": "SNS"
}]
```

Figure 2. Example of Needs Extraction API output

```
[{
"where": "everywhere",
"when": "every day",
"who": "I",
"whom": "people who match my interests",
"why": "",
"what": "connect and share real-time content",
"how": "SNS"
}]
```

Figure 3. Example of Need Restoration API input

Two values are input into the Needs Extraction API. The first is the user's need statement in natural language directed towards a virtual agent, specifying a particular service. The second is a list of service candidates for the how (desired service), derived from preliminary string searches on the need statement. The output yields the 6W1H need data extracted from the need statement in a JavaScript Object Notation (JSON) array format. For instance, when inputting the need statement "I want to connect with people who match my interests on SNS and share real-time content" and providing "SNS" as the how candidate, a result similar to Figure 2 can be expected. The output is not a single JSON format, but a JSON array, which can extract multiple 6W1H JSON elements corresponding to multiple needs related to a service or different services from a single utterance.

The Needs Re-extraction API requires three input values. The first two, the need statement and the how candidate list, are the same as the Needs Extraction API. The third is a previously extracted single 6W1H need data. The output merges the content of the need statement and the previous 6W1H need data, producing a new 6W1H need element in a JSON array format. If, for instance, the provided 6W1H need data lacks where and when elements, and the accompanying statement is "I want to use it every day in the living room", the output might include where as "living room" and when as "every day". However, when multiple 6W1H need data are output, it is necessary to determine if each 6W1H need data updates the input need data or is entirely new.

One potential method for this determination is considering the JSON as a string and identifying the updated need by the shortest Levenshtein distance.

For the Need Restoration API, the only input is the 6W1H need data. The output is a natural language sentence representing the need. For instance, when the 6W1H need data shown in Figure 3 is input, the output might be “Every day, I want to use SNS to connect with people who match my interests and to share real-time content anywhere.”

3.3. (A2) Design of the overall architecture

The overall architecture of the interactive user needs extraction system utilizing LLM is illustrated in Figure 4. The frontend application consists of a virtual dialogue agent and the Google Speech API [16]. The backend application is composed by integrating various systems and APIs, with the interactive needs extraction system at its core. These include the user management system, service management system, Needs Extraction API, Need Restoration API, and Chat API. The 6W1H need data extracted by the system is stored in the needs database through the interactive needs extraction system.

The system’s operations are executed in the order indicated by the arrows in Figure 4. Initially, a user vocalizes their service needs. This voice is recorded by the dialogue agent. The recorded voice is then converted into a text-based needs statement through the Google Speech API. Subsequently, this statement is passed to the interactive needs extraction system. The interactive needs extraction system first queries the user management system to determine whether the user who expressed the need is registered in the system. The text of the expressed need is normalized by converting full-width characters into half-width characters and lowercase letters into uppercase. In case of Japanese, kanji and hiragana characters into katakana are normalized to katakana characters. By comparing the normalized string with the list of registered service names retrieved from the service management system, candidates for the how (service name) that the need targets are extracted. The original expressed need and the how candidates are then fed into the Needs Extraction API or the Needs Re-extraction API, producing the 6W1H needs data as output. The needs extraction (or re-extraction) API internally invokes the Chat API using the GPT model. The extracted 6W1H needs data is accumulated in the MySQL database termed the needs database.

The process for generating a response is intricate, branching according to the content of the expressed need and the results of the needs extraction. To summarize, when a need statement is required for purposes like confirming a need with the user, the 6W1H need data is provided as input to the Need Restoration API, which outputs the need statement. Following that, conditional branching occurs based on the content of the 6W1H need data and the need statement, formulating the response. This generated response is handed over to the dialogue agent. Ultimately, the dialogue agent articulates the response to the user.

3.4. (A3) Design of the dialogue flow

There are two types of dialogue flows: (D1) a response to a new needs statement and (D2) a response to an additional statement made during needs extraction. A single cycle refers to the repetition of dialogue needed to extract one need related to a service. The (D1) flow is applied to the initial user statement of the cycle, whereas the (D2) flow is applied o user statements from the second instance onward. The operational steps for the (D1) response flow to a new needs statement are as follows:

Step 1. Extract the 6W1H need data from the received statement using the Needs Extraction API.

Step 2. If any 6W1H elements are missing, request additional information from the user.
Step 3. If no 6W1H elements are missing, restore the need statement using the Need Restoration API and confirm with the user if the need has been correctly extracted.

In the (D2) response flow to an additional statement during needs extraction, the same 6W1H need data can be continuously updated using a common dialogue ID to search the needs database. The operational steps for this response flow are as follows:

Step 1. Retrieve previously extracted 6W1H need data from the database based on the dialogue ID.

Step 2. Re-extract new 6W1H elements from the received additional statement and previous 6W1H need data using the Needs Re-extraction API.

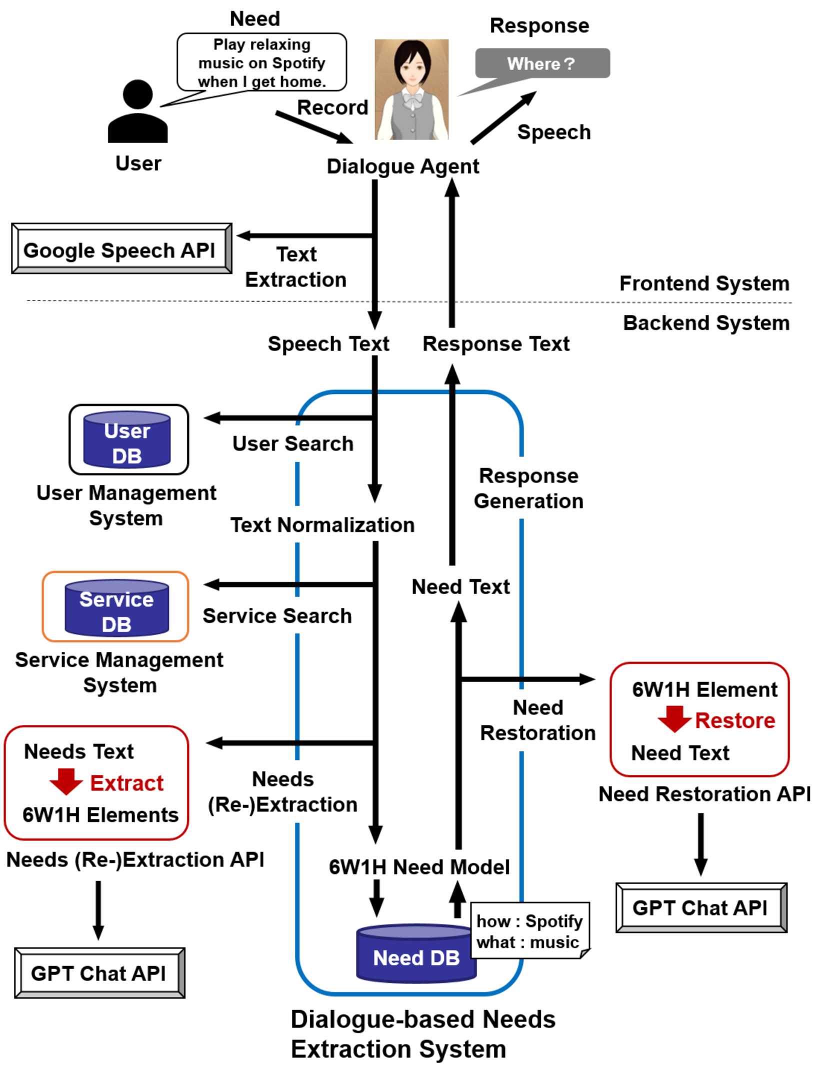

Figure 4. Overall architecture and operational flow of the needs extraction system

Step 3. If multiple 6W1H need data are re-extracted, determine the 6W1H need data closest to the previous 6W1H need data in terms of the Levenshtein distance through calculation. Consider this determined 6W1H need data as the updated data for the current dialogue.

Step 4. Generate the response statement in the same manner as Steps 2 and 3 of the (D1) flow.

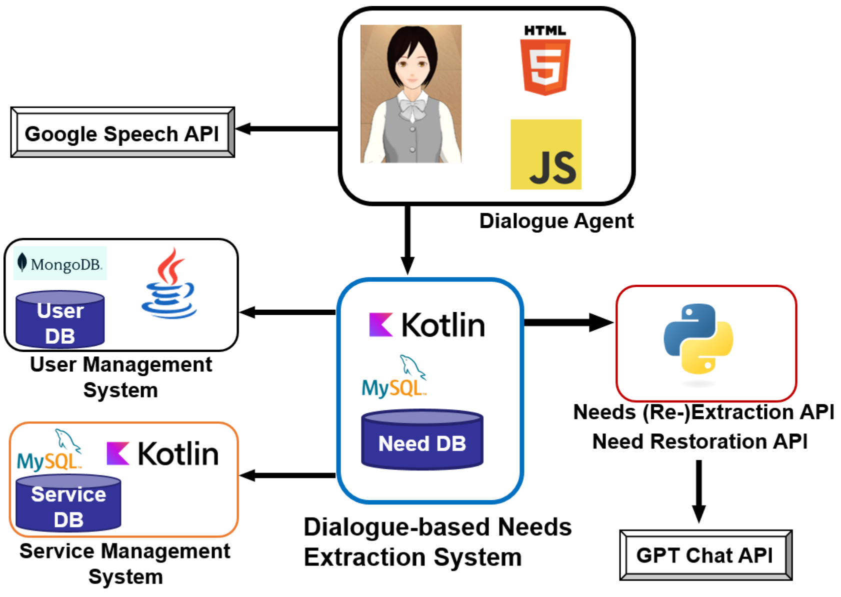

Figure 5. Implementation architecture of needs extraction system

4. Implementation

The implementation architecture is depicted in Figure 5. The Dialogue Agent is implemented using HyperText Markup Language (HTML) and JavaScript, functioning within a browser. The Dialogue-based Needs Extraction System is developed in Kotlin, with the Needs database realized using MySQL. The User Management System is implemented in Java, and rather than using MySQL, MongoDB is employed for the User database to represent complex attributes for each user in JSON. This facilitates the cross-utilization of the database across multiple systems, not limited to the proposed system. However, it is also feasible to implement the database using MySQL if user management is confined within the proposed system. The Service Management System is developed in Kotlin, with the Service database realized using MySQL. The Needs Extraction, Re-extraction, and Restoration API have a simple configuration, consisting solely of sending commands to the GPT Chat API, and are thus implemented on the same Python server. In this implementation, five servers are utilized to clarify roles. This allows for flexible configurations, such as installing servers handling personal information on-premises while placing others in the cloud. Alternatively, given that complex computations are performed using external services via the API, it is possible to virtually realize all five servers on a single server.

5. Needs Extraction & Restoration API Evaluation Experiment

We conducted an evaluation experiment with the objective of addressing the following research questions concerning the needs extraction and restoration API of (A1):

RQ1 Do the Needs Extraction API and Need Restoration API accurately and comprehensively convert needs?
RQ2 Does the Needs Extraction API sufficiently extract the 6W1H elements?

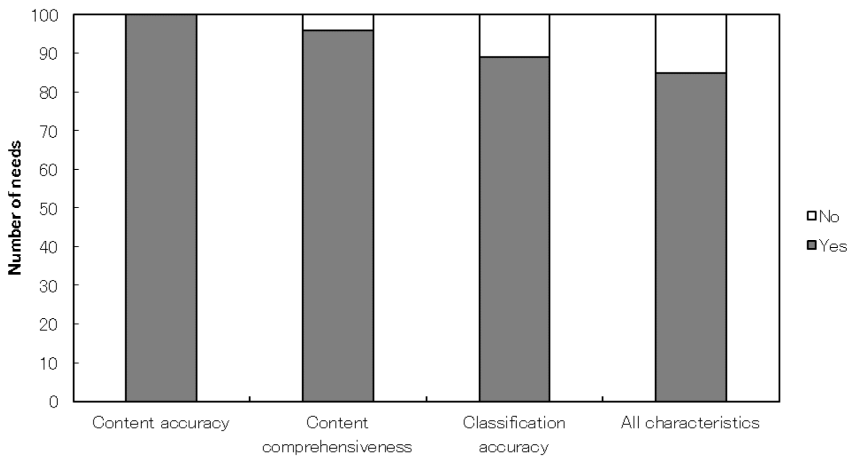

Figure 6. Characteristics of extraction results from Needs Extraction API

5.1. Experimental Setup

In the evaluation experiment for the Needs Extraction API, we generated test data for input to the API using ChatGPT. We created 100 variations of need statements, assuming various perspectives such as students, elderly, and working professionals using ChatGPT. These need statements encompassed a total of 28 different services. To examine the basic quality of the API, each need statement focused on only one service, and we refrained from using need statements that targeted multiple services. The evaluation involved authors analyzing the output results of the API, examining its characteristics and conditions. For the evaluation experiment of the Need Restoration API, we generated 6W1H need data as test input for the API using ChatGPT. The 100 variations of 6W1H need data were crafted by ChatGPT, taking into consideration various perspectives like students, elderly, and working professionals. The how element of the 6W1H need data included a total of 21 different services. The evaluation process consisted of the authors analysing the output results of the API to determine its characteristics.

5.2. Experimental Result

The results of the evaluation experiment for the Needs Extraction API are shown in Figures 6 and 7. The outcomes presented in Figure 6 are analyses related to three characteristics. Content accuracy indicates whether the extracted results contain any false information. Content comprehensiveness signifies whether all crucial information from the spoken sentence has been extracted. Classification accuracy represents whether each extracted need element has been appropriately categorized into one of the 6W1H elements. For instance, if an element, which should be extracted under where, such as “living room”, is categorized under when, then the output result was evaluated as not meeting classification accuracy. The graph also illustrates output results that simultaneously satisfy all characteristics.

The results in Figure 7 provide an analysis of how many types of the 6W1H elements could be extracted. In the section on detailed conditions, we analyzed whether at least one of the where, when, who, or whom elements, which express detailed conditions for service execution, has been extracted. In the section on overall conditions, we assessed whether the why element, which conveys the general conditions for service execution in a sentence, was extracted. The graph also displays the output results that extracted both detailed and general conditions, and those that extracted either one of the two.

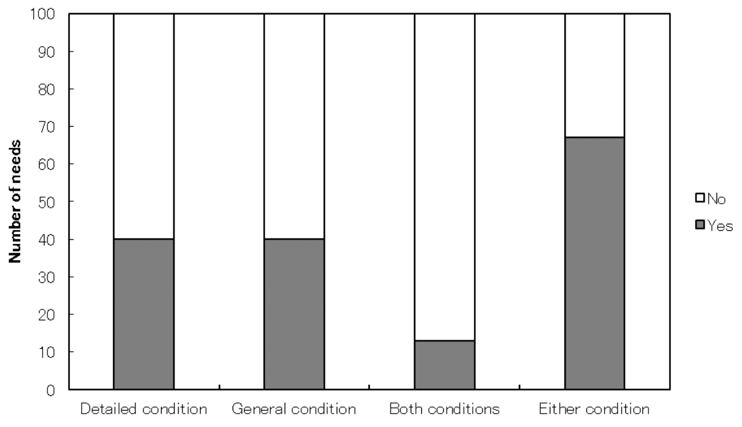

Figure 7. Conditions extracted by the Needs Extraction API

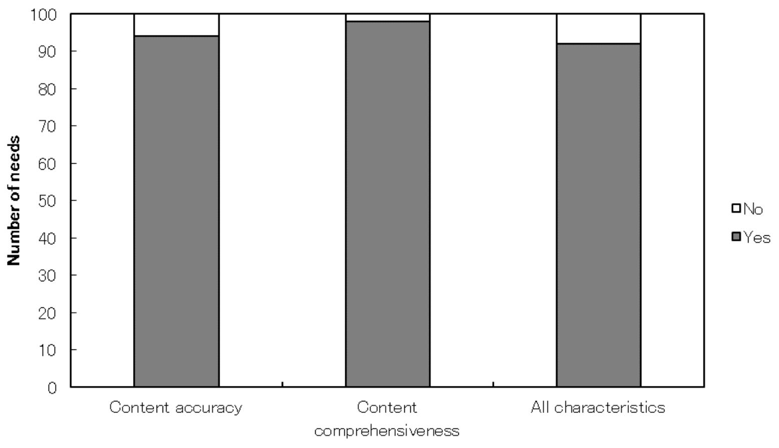

Figure 8. Characteristics of restoration results from Need Restoration API

The results of the evaluation experiment for the Need Restoration API are displayed in Figure 8. Content accuracy indicates whether the restored results contain any false information. Content comprehensiveness signifies whether all essential information from the 6W1H elements has been covered. The graph also illustrates output results that simultaneously satisfy all characteristics.

5.3. Experimental Discussion

We will discuss the results of the experiment with respect to RQ1 and RQ2. First, considering RQ1, we analyze whether the conversion of needs was carried out accurately and comprehensively. In the Needs Extraction API, the content accuracy was 100%, with the extraction results being entirely truthful and accurate. The content comprehensiveness stood at 96%. However, the omitted content was non-core, supplemental information such as “want to shop conveniently” or “wish to have a fun time”. The classification accuracy was 89%. There were many instances where content that should be categorized under detailed conditions like when or under what was mistakenly categorized under why. In the Need Restoration API, the content accuracy was 94%, and there were instances where the content of why and what was reversed in the formulation of sentences. The content comprehensiveness was 98%, and when multiple elements had similar content, some information to be formulated into sentences was missing. Furthermore, 85% in the extraction API and 92% in the restoration API satisfied all the characteristics. From these results, it can be said that in use cases where users verify the correct extraction of their needs through an interactive needs extraction system, the needs extraction and restoration API possess high accuracy and comprehensiveness.

Next, concerning RQ2, we consider the sufficiency of the 6W1H element extraction. In the Needs Extraction API, 40% could extract detailed conditions, and 40% could extract general conditions. Moreover, 67% could extract either the detailed conditions or the general conditions. From this, it is challenging to claim that sufficient 6W1H elements are extracted in a single extraction. However, the types of 6W1H elements that can be extracted from the need sentence depend on the content of the needs sentence. Furthermore, in use cases where the user is asked about the missing 6W1H conditions during repeated interactions, it can be said that the Needs Extraction API possesses the capability to extract the 6W1H elements adequately.

6. DISCUSSION

By utilizing the LLM for the mutual conversion between needs statements and 6W1H need data, we succeeded in extracting needs from complex natural language utterances. Specifically, we can now extract multiple needs from a single needs statement and have become able to extract the who and why elements, which were challenging in previous research. From the results of the API evaluation experiment, it was determined that needs can be accurately extracted with a single extraction. However, the characteristics regarding the classification and diversity of each element remain uncertain, highlighting the importance of dialogues that inquire about missing 6W1H elements. To realize a needs extraction dialogue, we leveraged a user search system, service search system, and needs extraction and restoration API, implementing the needs extraction system as a distributed microservices architecture. As future challenges, we believe it is essential to undertake performance evaluation experiments of the entire system and evaluate the system quality when used by actual users.

7. CONCLUSION

The objective of this research is to construct a method to extract 6W1H need data from complex natural language needs statements through dialogue with virtual agents, thereby enhancing the diversity and accuracy of needs extraction. As a key idea, we have developed a method to extract 6W1H need data from complex natural language needs statements using the LLM. Specifically, we designed and developed three mutual conversion APIs for extracting, re-extracting, and restoring 6W1H need data from natural language using the LLM. Moreover, we designed the overall architecture of the dialogue-based needs extraction system using the proposed APIs. By incorporating the LLM into the API, we have streamlined and redesigned the complex dialogue flow.

Additionally, we conducted evaluation experiments by inputting 100 test data into the Needs Extraction API and the Need Restoration API. As a result, the extraction API yielded outputs that satisfied 85% for accuracy and comprehensiveness, while the restoration API yielded 92%. Through this research, we have enabled the extraction of 6W1H need data from complex natural language with high accuracy and comprehensiveness using the LLM. For future research, we are considering evaluating the dialogue-based needs extraction system in more practical applications.

ACKNOWLEDGEMENTS

This research was partially supported by JSPS KAKENHI Grant Numbers JP19H01138, JP20H05706, JP20H04014, JP20K11059, JP22H03699, JP19K02973, and Young Scientists (No.23K17006)

REFERENCES

[1] Fan, H. & Poole, M.S., (2006) "What is personalization? perspectives on the design and implementation of personalization in information systems," *Journal of Organizational Computing and Electronic Commerce*, Vol.16, No.3-4, pp179-202.

[2] Hammi, B., Zeadally, S., Khatoun, R., & Nebhen, J., (2022) "Survey on smart homes: Vulnerabilities, risks, and countermeasures," *Computers & Security*, Vol.117, pp102677.

[3] Huang, J., Tong, Z., & Feng, Z., (2022) "Geographical POI recommendation for Internet of Things: A federated learning approach using matrix factorization," *International Journal of Communication Systems*, ppe5161.

[4] Jannach, D., Manzoor, A., Cai, W., & Chen, L., (2021) "A survey on conversational recommender systems," *ACM Computing Surveys*, Vol.54, pp1-36.

[5] Hollis, K., Soualmia, L., & S´eroussi, B., (2019) "Artificial intelligence in health informatics: Hype or reality?," *Yearbook of Medical Informatics*, Vol.28, pp3-4.

[6] Zhang, Y. & Chen, X., (2020) "Explainable Recommendation: A Survey and New Perspectives," *Foundations and Trends® in Information Retrieval*, Vol.14, No.1, pp.1-101.

[7] Nakata, T., Chen, S., Saiki, S., & Nakamura, M., (2023) "Dialogue-Based User Needs Extraction for Effective Service Personalization," *HIMI 2023, Held as Part of the 25th HCI International Conference, HCII 2023*, Vol.LNCS 14016, pp139-153.

[8] Cascella, M., Montomoli, J., Bellini, V., & Bignami, E., (2023) "Evaluating the feasibility of chatgpt in healthcare: An analysis of multiple clinical and research scenarios," *Journal of Medical Systems*, Vol.47, No.1, pp33.

[9] Rostami, M., Oussalah, M., & Farrahi, V., (2022) "A Novel Time-Aware Food Recommender-System Based on Deep Learning and Graph Clustering," *IEEE Access*, Vol.10, pp52508-52524.

[10] Goldenberg, D., Kofman, K., Albert, J., Mizrachi, S., Horowitz, A., & Teinemaa, I., (2021) "Personalization in practice: Methods and applications," *WSDM '21*, pp1123-1126.

[11] Kocaballi, A.B., Berkovsky, S., Quiroz, J., Laranjo, L., Tong, H.L., Rezazadegan, D., Briatore, A., & Coiera, E., (2019) "The personalization of conversational agents in health care: Systematic review," *Journal of Medical Internet Research*, Vol.21, ppe15360.

[12] Ozono, H., Chen, S., & Nakamura, M., (2022) "Encouraging elderly self-care by integrating speech dialogue agent and wearable device," *8th International Conference, ITAP 2022, Held as Part of the 24th HCI International Conference, HCII 2022*, Vol.LNCS 13331, pp52-70.

[13] Tinn, R., Cheng, H., Gu, Y., Usuyama, N., Liu, X., Naumann, T., Gao, J., & Poon, H., (2023). "Fine-tuning large neural language models for biomedical natural language processing," *Patterns*, Vol.4, No.4, pp100729.

[14] Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M.T., & Zhang, Y., (2023) "Sparks of artificial general intelligence: Early experiments with gpt-4," ArXiv, Vol.abs/2303, pp12712.

[15] Lu, Y., Bartolo, M., Moore, A., Riedel, S., & Stenetorp, P., (2022) "Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Vol.1: Long Papers, pp8086-8098.

[16] Speech-to-Text: Automatic Speech Recognition — Google Cloud. https://cloud.google.com/speech-to-text?hl=en. [Online]. (Accessed on 10/18/2023).

AUTHORS

Takuya Nakata received Master of System Informatics from Kobe University, and he did Bachelor degree in Engineering. Currently, he is pursuing his PhD in Engineering from Kobe University. His research interests include Service computing, Software engineering.

Sinan Chen received a Master of system informatics and a Ph.D. in computational science from Kobe University, Japan, in 2019 and 2021, respectively. He worked at Kobe University as a research fellow (PD) from the Japan Society for the Promotion of Science (JSPS) in the first half of 2022. He is currently an assistant professor in the Center of Mathematical and Data Science Center at Kobe University, Japan. His research interests include the smart home, cloud computing, machine learning, and software engineering. He is a member of the IEEE, ACM, Sigma Xi, CAAI, CCF, IEICE, and IEEJ.

Masahide Nakamura received the B.E., M.E., and Ph.D. degrees in Information and Computer Sciences from Osaka University, Japan, in 1994, 1996, 1999, respectively. From 1999 to 2000, he has been a post-doctoral fellow in SITE at University of Ottawa, Canada. He joined Cybermedia Center at Osaka University from 2000 to 2002. From 2002 to 2007, he worked for the Graduate School of Information Science at Nara Institute of Science and Technology, Japan. From 2007 to 2022, he worked for the Graduate School of System Informatics at Kobe University. He is currently a full professor in the Center of Mathematical and Data Science Center at Kobe University. In 2015, he worked for Universite Grenoble Alpes as a visiting professor. In 2018, he joined RIKEN Center for Advanced Intelligence Project as a visiting researcher. His research interests include the service/cloud computing, smart home, smart city, and gerontechnology. He is a member of the IEEE, ACM, IEICE and IPSJ.

© 2023 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.