Regular Paper

Efficient Classification of Non-Functional Requirements Using ChatGPT's Function Calling Feature

Kazuhiro Mukaida*, Seiji Fukui** , Takeshi Nagaoka** , Takayuki Kitagawa ** , Shinpei Ogata* and Kozo Okano*

*Graduate School of Science and Technology, Shinshu University, Japan
**Toshiba Corporation, Japan
24hs254c@shinshu-u.ac.jp, {Fukui.Seiji, Nagaoka.Takeshi, Kitagawa.Takayuki}@toshiba-sol.co.jp, {ogata, okano}@cs.shinshu-u.ac.jp

Abstract - Efficient analysis of requirement specifications is crucial for improving their quality, which is important in software development. We focus on non-functional requirements (NFRs), which are often overlooked in requirement definitions of system developments and propose a method that allows individuals without extensive expertise to efficiently extract and classify NFRs from requirement specifications. The proposed method aims to efficiently enhance the quality of software requirement specifications by enabling the extraction and classification of NFRs with minimal expertise.

Previously, the authors experimented with creating dedicated deep learning models for classification tasks and then used pre-trained Transformer models like BERT and GPT-2, trained on human-annotated datasets. However, recent advancements, such as tools like ChatGPT, enable classification via prompt interactions alone. In this paper, we explore the capabilities of ChatGPT's Function Calling feature, utilizing an approach that optimizes its behavior, aiming to demonstrate its superior classification performance compared to prompt-only responses and conventional classification methods, which require large training datasets.

Function Calling significantly reduced ambiguities and improved classification accuracy by ensuring adherence to predefined classification boundaries. For example, GPT-4 Turbo demonstrated an F1 score improvement from 0.681 to 0.753, and GPT-3.5 Turbo achieved an increase from 0.587 to 0.651. Additionally, GPT-4o showed a gain from 0.754 to 0.780. These improvements highlight the practical utility of Function Calling as a primary classification tool.

As supplementary verification, we conducted two additional analyses. Fine-tuning GPT-3.5 Turbo on small datasets significantly enhanced its performance, achieving an F1 score of 0.796. Similarly, incorporating sentence concatenation by linking preceding and following sentences improved contextual understanding, increasing accuracy from 0.792 to 0.831. These approaches, while complementary, further validated the robustness of Function Calling for NFR classification tasks.

Future research should address remaining challenges, such as improving the model's contextual understanding and developing targeted training datasets that emphasize the most challenging classification categories. These findings highlight the potential of advanced natural language models like ChatGPT in making NFR classification more efficient and precise.

Keywords: Function Calling, ChatGPT, GPT-4o, Non-Functional Requirements, Documents Classification

1 INTRODUCTION

In information system development, requirements are broadly divided into functional and non-functional requirements [1]. Functional requirements define specific functions that the system must perform, whereas non-functional requirements describe the overall qualities of the system, such as availability, performance, reliability, and efficiency. These non-functional requirements define expectations for the system's actual operating environment and are crucial for ensuring the overall quality of the software [2].

1.1 Importance and Challenges of Non-Functional Requirements

Non-functional requirements are critically important for the success of a system, but their abstract nature makes their identification and classification challenging. If non-functional requirements are not adequately defined, they can severely impact system performance, usability, and security. Therefore, clearly defining non-functional requirements and managing them throughout the development process is one of the major challenges in system development. Insufficient non-functional requirements can result in the system failing to meet expected performance, ultimately leading to decreased user satisfaction.

Non-functional requirements are essential for ensuring the overall quality of the system. Characteristics such as system response time, throughput, availability, security, usability, and scalability directly affect user experience and reliability. For example, if a system has a slow response time, users may find it difficult to use and avoid using it altogether. Additionally, if security requirements are not adequately met, the system may be vulnerable to external attacks, increasing the risk of important data being leaked. Thus, non-functional requirements are vital for maintaining the health of the system and the trust of users.

To properly define and agree on non-functional requirements, it is necessary to clearly identify and classify

them. However, since non-functional requirements are abstract and diverse, manual classification is time-consuming, labor-intensive, and requires specialized knowledge. Disparities in understanding non-functional requirements between users and vendors often arise, becoming an obstacle to the success of system development.

1.2 Emergence and Advancement of Automated Classification Techniques

As a means of addressing the challenges of identifying and classifying non-functional requirements, automated classification techniques are gaining attention. In particular, research on document classification techniques using deep learning and large language models (LLMs) has been actively conducted. Since the advent of Transformer by Vaswani et al. [3], models such as BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. [4] and GPT (Generative Pre-trained Transformer) by Radford et al. [5], which emerged in 2018, have been pre-trained on large amounts of text and fine-tuned for specific tasks to improve classification accuracy. The authors have also experimented with automatic classification using a model incorporating BERT and GPT-2 [6] in the previous paper [7].

Given the abstract nature and diversity of non-functional requirements, the introduction of automated classification techniques is highly beneficial. First, defining and classifying non-functional requirements require the expertise and time of experienced professionals, making manual classification costly and time-consuming. Automated classification techniques significantly reduce these manual efforts, allowing for efficient classification. Moreover, automated classification provides consistent results and reduces human errors and biases. Additionally, accurate and swift classification of non-functional requirements enables appropriate requirements management from the early stages of a development project, thereby improving the project's success rate. For instance, properly classifying and recognizing system performance requirements early on can help prevent performance issues in later development stages.

Recently, with the widespread use of conversational models like ChatGPT, inference through prompt interactions is becoming possible. Conversational LLM models generate appropriate responses in response to questions and instructions during interaction with users in natural language. This conversational capability allows users to perform advanced inferences and information searches without requiring specific knowledge or skills.

Models generate responses based on prompts. Thus, by designing prompts appropriately, the output of the model can be controlled to obtain responses suitable for the intended purpose. For example, the format and content of prompts can be adjusted to handle various tasks, such as question answering, text generation, summarization, and translation.

However, there are limitations to performance in classification tasks. Classification tasks involve categorizing text or data into specific categories or labels, which is one of the fundamental applications of deep learning models. For instance, detecting spam emails or categorizing product reviews as positive or negative are common classification tasks. While the classification performance of conversational models has improved, they may still lag behind dedicated traditional models for certain complex tasks or large datasets. Particularly, when handling multi-class classification beyond binary classification, hallucinations may occur, resulting in the unintended creation of classification categories, which negatively impacts accuracy.

In this paper, we attempt to classify non-functional requirements using the Function Calling feature of ChatGPT. Function Calling is a means of obtaining additional information by calling other APIs or functions during prompt interactions, allowing integration with external services and databases. Generally, by using this feature, more advanced processing and data retrieval become possible.

This paper focuses on utilizing arguments required for calls during the Function Calling process as classification data. By employing enum to restrict and enumerate variable types, unintended category creation due to hallucinations can be eliminated, ensuring that the classification remains within the intended boundaries. The model automatically suggests appropriate arguments based on contextual information, and proper argument content is directly linked to accurate calls and responses. Utilizing these argument suggestions as classification data is expected to improve the classification accuracy of non-functional requirements.

In addition to the core focus on the outcomes of Function Calling, the following two experiments were supplementary investigations aimed at further enhancing the effectiveness of the Function Calling feature. While these experiments provide valuable insights into potential improvements, the primary objective of this paper remains centered on the results derived from Function Calling.

First, we verify the impact of small-scale fine-tuning using non-functional Requirements Grades [8] on classification by Function Calling. By optimizing the Function Calling feature of ChatGPT based on clearly defined standards using non-functional Requirements Grades, we aim to improve the classification accuracy of non-functional requirements. The outcomes of this study are expected to contribute to the development of automated classification technologies for non-functional requirements and enhance the efficiency of the quality assurance process in system development.

Second, we propose a classification method that uses sentence concatenation to provide additional contextual information. In conventional classification methods, only the target sentence is classified, but incorporating contextual information can improve classification accuracy. In requirement specifications, similar non-functional requirements may span multiple sentences, and the preceding and following sentences are often useful as additional contextual information for the target sentence. In this method, we embed the test data, determine similarity based on cosine similarity between the target sentence and its surrounding sentences, and concatenate related sentences before classification. This allows for classification that

considers contextual information, thereby improving classification accuracy.

In the future, automated classification technologies for non-functional requirements are expected to evolve further, enabling more accurate and efficient classification. In particular, with the advancement of large language models, it will become possible to understand the abstract nature of non-functional requirements more deeply and perform classification accordingly.

2 RELATED RESEARCH

2.1 Classification of Non-Functional Requirements

Kinoshita et al. [9] proposed a method for extracting non-functional requirements (NFRs) from Japanese requirement documents. Specifically, they established keywords related to NFRs and applied them to the documents to effectively identify relevant requirements. Additionally, they introduced an approach for detecting errors in NFRs by defining case frames for the actions described in requirement sentences. Their method involves validating these sentences using criteria such as omission, ambiguity, redundancy, and inconsistency, thereby enhancing the overall quality of the requirements.

2.2 Classification Using Deep Learning

Gnanasekaran et al. [10] researched and developed recurrent neural network (RNN) models, known for their effectiveness in processing sequential natural language text. They conducted experiments using these models to classify NFRs described in natural language into five categories: maintainability, operability, performance, security, and usability. The experimental validation was based on two datasets encompassing approximately 1,000 NFRs, demonstrating the potential of RNNs in accurately classifying NFRs.

Kitagawa and Nagaoka [11] proposed an automatic classification method using Word2Vec and convolutional neural networks (CNN). Their approach extended the classification targets beyond NFRs and project management-related sentences to include functional requirements and other types of sentences. They aimed to automatically classify all sentences in Japanese Request for Proposal (RFP) documents, improving the efficiency and accuracy of requirement specification analysis.

2.3 Classification Using Large Language Models

Since the introduction of the Transformer architecture, significant advancements have been made in natural language processing tasks using LLMs based on Transformers. These models have largely replaced traditional recurrent neural networks such as RNNs, LSTMs, and GRUs. By fine-tuning large pre-trained models like BERT [4] for specific tasks, researchers have achieved models optimized for those tasks, often attaining higher accuracy.

Zhu et al. [12], starting from BERT, introduced a novel method to enhance classification performance by formalizing input sentences as natural language templates and leveraging knowledge expansion [13]. This approach improved short text classification by integrating additional contextual information into the prompts.

Gutierrez et al. [14] reported that fine-tuned BERT models outperformed interactive GPT-3 models in tasks such as named entity recognition and relation extraction within the medical domain. Similarly, Sun et al. [15] found that while interactive models like GPT-3 can perform text classification tasks, fine-tuned task-specific models still achieve higher accuracy due to their specialized training.

Ibe et al. [16] conducted experiments using large generative language models to automatically classify requirement specifications with few examples by adjusting prompts. Their results indicated that BERT-based classifications outperformed interactive models like ChatGPT in terms of accuracy, highlighting the effectiveness of task-specific fine-tuning.

On the other hand, the performance of ChatGPT as a question-answering system (QAS) has been reported to match or even exceed that of traditional task-specific QAS models. Tan et al. [17] demonstrated that ChatGPT provides superior robustness and explainability in conversations compared to traditional QAS, offering enhanced user interaction and adaptability.

Brown et al. [18] evaluated GPT-3's performance in few-shot learning settings, reporting high effectiveness in natural language processing tasks under zero-shot and one-shot conditions. Their work highlighted the capability of large language models to perform various tasks without extensive task-specific fine-tuning, showcasing the potential of models like GPT-3 in few-shot learning scenarios.

In studies focusing on prompt engineering, the importance of In-Context Learning (ICL) has been emphasized. Dong et al. [19] provided a comprehensive overview of ICL, demonstrating its applicability across diverse tasks and its ability to improve model performance by incorporating contextual information during inference. Liu and Yang [13] further explored knowledge-enhanced prompt learning for few-shot text classification, showing that integrating external knowledge can significantly boost performance.

Min et al. [20] analyzed how the selection of demonstrations in ICL affects model performance, clarifying the optimal conditions for demonstration selection to maximize effectiveness. Wan et al. [21] showcased the effectiveness of ICL in relation to extraction tasks, proposing methods with higher flexibility and accuracy compared to conventional approaches by utilizing contextual cues within the prompts. These studies highlight the flexibility and performance of interactive models and ICL across various tasks, opening new possibilities in the field of natural language processing.

3 PROPOSED METHOD

This paper proposes an automatic classification method for NFRs in system development specification documents. This method leverages the Function Calling feature of GPT models, a type of large language model, and is based on IPA standards.

3.1 Function Calling

Function Calling is a feature provided by the API of large language models, enabling the model to indirectly interact with external APIs or systems and generate specific actions. This technology involves parsing natural language queries, selecting appropriate functions, and generating JSON responses. For example, in response to the query “Tell me about the weather in Tokyo,” the model suggests calling a function that uses a weather API with the region name as an argument. The argument "Tokyo" is prepared for the weather API call. The program then retrieves Tokyo's weather through the function and weather API, then passes this information back to the model as an additional prompt. The model then provides the final answer, significantly expanding the potential for interactive applications and services using the model. This feature bridges the gap between natural language understanding and the execution of specific actions, enhancing the model's practicality and allowing it to flexibly respond to user requests. Function Calling not only streamlines the interaction between the model and external systems but also improves the overall efficiency of handling user queries by automating the process of converting natural language into actionable commands.

3.2 Classification using Function Call

A derivative use of Function Calling involves extracting structured data from text. For example, a function like extract_data(name: string, birthday: string) can be defined and invoked as needed, extracting a person's name and birthday from the text as arguments. This functionality allows for the rapid and accurate extraction of necessary information from large volumes of text data, facilitating efficient data analysis and information management. This method provides a more structured and organized approach to handling text data, ensuring that the extracted information is consistent and accurate.

Function Calling can also help in selecting the most appropriate argument based on the overall context of a query, even if the specific term is not explicitly mentioned. By leveraging the model's natural language understanding, it can infer the intended meaning and relevant information from the user's input.

In this study, rather than using Function Calling for extracting structured data from documents, we applied it to document classification. Normally, Function Calling in LLMs is triggered only when the model determines it is necessary. However, we configured parameters to ensure that Function Calling is always invoked whenever a classification target sentence is input. This guarantees that Function Calling is consistently performed during classification tasks.

The function invoked in this process exists only as a description of its name and outline and does not exist as an actual implemented program. The primary purpose of Function Calling in our method is to return arguments that indicate classification categories. These arguments are subsequently used within the program for further processing. Since the goal of this experiment is to acquire the necessary arguments for classification, the Function Calling process terminates once the arguments are obtained, without invoking any external modules or APIs. This ensures that the classification process benefits from the structure and consistency of Function Calling without relying on external integrations.

To achieve classification, we enforced the invocation of a dummy function through prompts containing evaluation sentences. The model suggests classification categories as arguments during this process. By specifying the argument type as enum, we enumerated the classification categories, limiting suggestions to predefined options and preventing hallucinations. This approach automates and streamlines the classification process, reducing the potential for human error and enhancing overall efficiency.

3.3 Supplemental Analysis 1 Fine Tuning for GPT-3.5

Section 3.3 and 3.4 aim to enhance the results achieved by utilizing Function Calling through two additional techniques. The purpose of these experiments is to improve the classification accuracy of NFRs by fine-tuning and enhancing contextual understanding.

Firstly, this paper attempts to improve models with classification accuracy by additional training of GPT-3.5 Turbo with a small amount of training data. Fine-tuning enables the model to learn language expressions specific to NFRs tasks or domains. This helps the model understand specialized terminology and contexts that are challenging for general language models, achieving higher classification accuracy. Moreover, fine-tuning with a small amount of training data significantly reduces the time and cost associated with preparing annotated large datasets, compared to traditional methods. For complex tasks like NFRs classification, it is crucial to learn efficiently from limited examples.

Fine-tuning not only tailors the model to specific tasks but also enhances its ability to handle nuanced and domain-specific language, resulting in more precise and reliable classifications. This approach is feasible even in scenarios where annotated data is scarce, enabling effective learning and adaptation with minimal resources.

3.4 Supplemental Analysis 2 Enhancing Contextual Understanding

Lastly, this paper proposes the use of sentence concatenation to provide additional contextual information that improves classification accuracy. The classification model has traditionally focused only on the target sentence,

but incorporating contextual information can enhance performance by capturing the broader context in which the target sentence appears. In requirement specifications, sequential sentences often describe similar non-functional requirements, and the preceding and following sentences frequently contain valuable information that helps clarify the classification of the target sentence.

In this method, embeddings are generated for each sentence in the test data using the text-embedding-3-large model. This model, provided by OpenAI, is an advanced natural language processing model capable of converting sentence semantics into numerical vectors. The embeddings are 3072-dimensional vectors that capture the nuances of sentence content and context, enabling accurate similarity assessments. These representations are essential for various tasks, including classification, search, and semantic comparison.

To identify relevant context, cosine similarity is calculated between the target sentence and its preceding and following sentences. Cosine similarity measures the angle between two vectors, with values closer to 1 indicating higher similarity. When the cosine similarity exceeds 0.5, the adjacent sentence is considered contextually relevant and is concatenated with the target sentence. This process generates a new, enriched input that reflects not only the target sentence but also the surrounding context. For example, if there are consecutive sentences A, B, and C, and B is the classification target, and if A and B are similar while B and C are not, the classification is performed using a concatenated sentence consisting of A and B.

It is possible that increasing the amount of text through concatenation may enhance classification performance to some extent, even if the added text is not directly relevant. However, adding unrelated text can sometimes introduce noise and ambiguity, which may complicate classification rather than improve it. To address this, sentences are concatenated selectively based on their contextual relevance, as determined by cosine similarity. This approach helps ensure that the additional information enhances the classification process by reinforcing the semantic continuity of the text.

Table 1 Sources of Training Data by Organization

Organization	Project Description	Date
Ministry of Economy, Trade and Industry	Industrial Safety System Update	Jan-22
Digital Agency	Development and Operation/Maintenance Work for Account Information Registration Linkage System	Jul-22
Ministry of Health, Labor and Welfare	National Unified System for Medical Function Information Provision System and Pharmacy Function Information Provision System,	Jan-23

By concatenating relevant sentences, the model gains supplementary cues that contribute to more accurate classification. This method helps reduce the risk of misclassification by avoiding the inclusion of unrelated information and maintaining a focus on contextually aligned data.

The test data described in Section 4.1, derived from actual requirement specifications, was used for this experiment. Since the target sentences were extracted directly from these documents, the preceding and following sentences were naturally part of the same text, ensuring contextual relevance.

Experiments conducted with and without sentence concatenation demonstrated that improvements in classification accuracy were due to the inclusion of relevant contextual information, rather than merely increasing the amount of text. This structured approach highlights the value of selectively expanding context to enhance classification performance while minimizing the risk of unnecessary complexity.

4 EVALUATION EXPERIMENT

4.1 Models and Test Data Set

In this paper, we utilized the ChatGPT API with the following models:

GPT-4 Variants
- gpt-4o-2024-05-13
- gpt-4-turbo-2024-04-09

GPT-3.5 Turbo
- gpt-3.5-turbo-0125

The gpt-4o-2024-05-13 model represents the advanced current iteration of the GPT-4 series, incorporating the recent advancements in language understanding and generation capabilities.

The gpt-4-turbo-2024-04-09 model is offering optimized performance and a popular choice for a wide range of practical applications.

The gpt-3.5-turbo-0125 model, although from a previous generation, provides a valuable comparison point. It offers cost-effective performance and promptly supports fine-tuning, allowing for customization to specific tasks or domains. This makes the gpt-3.5-turbo-0125 model advantageous for scenarios where budget constraints and the need for tailored solutions are paramount.

Table 2 Number of Test Data Instances for Each Classification

Availability	81
Performance/Scalability	70
Operability/Maintainability	90
Migratability	60
Security	90
System Environment/Ecology	23
Total	414

In our analysis, we specified that the argument type for function calling properties is enum. For enum types, we defined "availability, performance/scalability, operability/maintainability, migratability, security, and system environment" based on the classification of non-functional requirements grades. By doing so, we were able to prevent hallucinations of unspecified categories.

Models provided responses for each evaluation sentence by sentence. The temperature was set to 0 to ensure deterministic outputs.

The primary focus of the experiment is to verify the superiority of using Function Calling capabilities over the baseline case, which relies solely on prompt interactions for classification. At the same time, we are able to observe the differences between models. This comparison provides valuable insights into their practical applications and efficiency, highlighting the strengths and possible improvements.

The test dataset used in this paper was constructed based on the existing requirement specifications of public tenders announced by government agencies shown in Table1. The requirement specifications were randomly selected. The test data was randomly extracted from these documents, and labeling was performed by us based on the IPA's non-functional requirements grades. The detailed distribution of sentences for each label is shown in Table 2.

4.2 Baseline Classification Method without Function Calling

To establish a baseline for classification accuracy, we evaluated a method that relies solely on prompt interactions, without utilizing the Function Calling feature. This baseline approach involved presenting the model with a prompt that defined the classification task and the sentence to be classified. The experiment was conducted in a zero-shot setting, with no examples provided to the model. This approach was chosen to evaluate the model's performance without prior contextual learning or fine tuning, thereby establishing a fair baseline for comparison.

The prompt used for this baseline is as follows:

"In the IPA's non-functional requirement grades, non-functional requirements are classified into six categories: availability, performance/scalability, operability/maintainability, migratability, security, and system environment. Answer with the name of only one of these six categories."

This prompt was consistently applied across both the prompt-only method and the Function Calling method to ensure a fair comparison. The purpose of the experiment was not to argue that the Function Calling method inherently surpasses the performance of the prompt-only method under varying conditions. Instead, the objective was to demonstrate that the addition of Function Calling to the same prompt leads to improved classification accuracy. By applying the same prompt in both cases, the comparison highlights the effect of integrating Function Calling, rather than differences in prompt design.

It is acknowledged that the application of advanced prompt engineering techniques or in-context learning could potentially enhance the performance of the prompt-only method. However, such approaches fall outside the scope of this study. The primary focus is to illustrate how the incorporation of Function Calling enhances performance, even when used alongside a standard, unoptimized prompt.

By establishing this baseline, the experiment aims to provide a clear and reproducible comparison between the two approaches, ensuring that the observed improvements in classification accuracy can be attributed directly to the inclusion of Function Calling, rather than variations in prompt formulation or tuning strategies.

4.3 Effect of Function Calling

Table 3 compares the classification results using Function Calling and prompt-only methods for each model. The evaluation metrics include accuracy, precision, recall, and F1-score, with the averages calculated using macro-averaging.

In most models, the method using Function Calling achieved higher accuracy compared to the prompt-only method. For instance, GPT-3.5 Turbo shows an accuracy of 0.705 with Function Calling versus 0.696 with prompt only. GPT-4 Turbo demonstrates even more significant gains with Function Calling, achieving 0.775 compared to 0.740 with prompt-only. The GPT-4o model also exhibits superior performance with Function Calling, reaching an accuracy of 0.792 compared to 0.783 with prompt only. For GPT-4o, the performance improvement with prompt-only methods is remarkable, reducing the advantage of Function Calling.

Table 3 Classification Performance Metrics for GPT-3.5 Turbo, GPT-4 Turbo, and GPT-4o using Function Calling and Prompt-Only Methods

	GPT-3.5 Turbo		GPT-4 Turbo		GPT-4o	
	Function Calling	Prompt only	Function Calling	Prompt only	Function Calling	Prompt only
Accuracy	0.705	0.696	0.775	0.740	0.792	0.783
Precision	0.719	0.608	0.811	0.766	0.785	0.787
Recall	0.648	0.618	0.734	0.674	0.779	0.741
F1-score	0.651	0.587	0.753	0.681	0.780	0.754

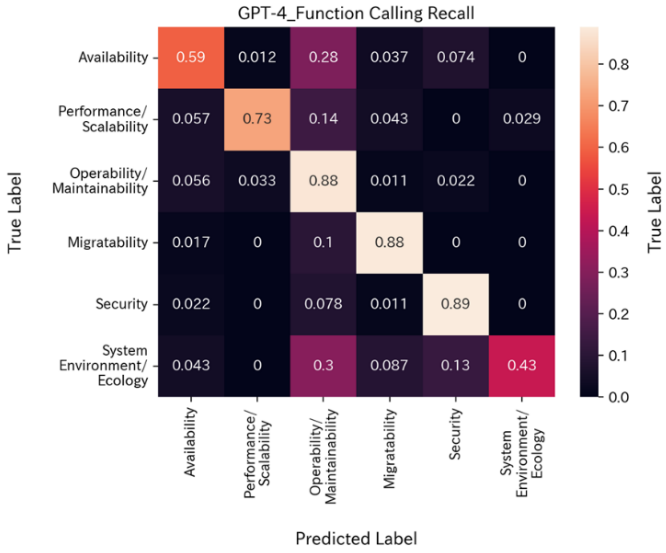

Figure 1 Recall Matrix for GPT-4 Turbo Function Calling and Prompting Performance Across 6 Categories

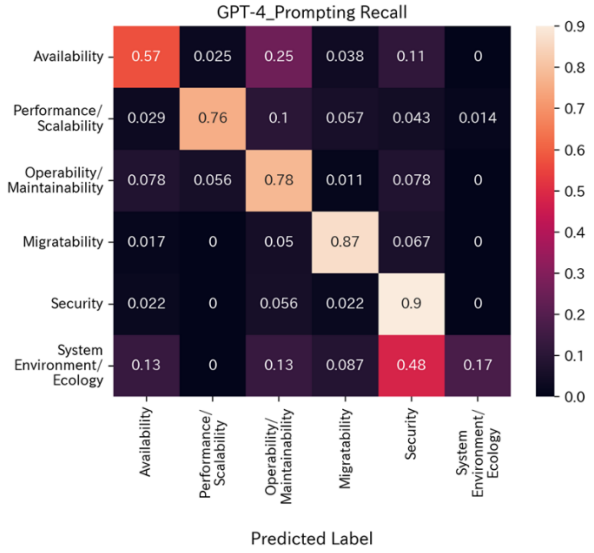

Figure 2 Recall Matrix for GPT-4 Turbo Prompting only Performance Across 6 Categories

Overall, these results suggest that the use of Function Calling significantly enhances the performance of language models across various evaluation metrics. While there are some nuances, particularly with GPT-4o, the general trend highlights the practical advantages of this approach in classification tasks.

4.4 Misclassification Analysis

In this section, we examine the misclassification issues observed across various categories, focusing on the "System Environment/Ecology" category. Figure 1 and Fig. 2 illustrate the recall values for Function Calling and prompting methods across categories, highlighting areas with pronounced misclassification.

Both Function Calling and prompting methods demonstrate low recall values for the "System Environment/Ecology" category, with prompting performing particularly poorly, likely due to the inherent complexity of this category. To illustrate the challenges and successes in classifying non-functional requirements, we present specific examples of correctly classified and misclassified cases within the "System Environment/Ecology" category.

The first example pertains to the System Environment category, "The construction environment for the next-generation security network assumes the use of the cloud, making physical servers and devices unnecessary." It was correctly classified as belonging to the System Environment category due to its explicit focus on operational premises and infrastructure design.

Conversely, the second example highlights a misclassification case. The requirement "Access to the facility and rooms housing managed devices is restricted, particularly for rooms containing devices, where only authorized personnel with management privileges are permitted entry" was incorrectly categorized as Security instead of System Environment. While this requirement involves access control measures, its primary focus is on the physical environment and the placement of managed devices, which aligns more closely with the System Environment category.

According to the IPA's non-functional requirements grade, the System Environment/Ecology category encompasses system constraints/premises established at installation, system characteristics, compliance standards, and environmental . associated with users and the region surrounding the system. Collectively, these elements represent the broader environmental and contextual factors influencing system operation and integration.

The low recall in this category likely stems from its broad and multifaceted nature, covering diverse aspects such as technical constraints and user characteristics. This complexity introduces ambiguity, making it difficult for models to distinguish relevant terms from other categories. Additionally, overlap exists between the "System Environment/Ecology" category and others; for instance, environmental conditions may also pertain to security concerns. Such overlaps complicate classification, as models often struggle to differentiate between closely related concepts. Moreover, interpreting terms related to the system environment is highly context-dependent, which hinders the models' ability to generalize accurately.

Overall, misclassification remains a critical challenge, particularly within the "System Environment/Ecology" category, due to its inherent complexity and the overlap with other classifications.

Table 4 Number of Training Data Instances for NFRs Grade

Availability	168
Performance/Scalability	175
Operability/Maintainability	158
Migratability	101
Security	163
System Environment/Ecology	168
Total	933

5 SUPPLEMENTAL EXPERIMENTS

5.1 Training Data for Fine Tuning and Embedding Analysis Fine Tuning Models

GPT-3.5 Turbo is capable of fine-tuning with even small amounts of data. Fine-tuning is greatly influenced by the quality of training dataset used, making its selection extremely important. In this paper, we utilized descriptions of non-functional requirement grades as the training dataset instead of using sentences from actual specifications. The primary reason for this choice is that it allows for more efficient annotation, as the classification of sentences is more straightforward. Additionally, one reason is the hypothesis that it may be difficult to learn the diversity of NFRs with a small amount of data with small numbers of training data. The non-functional requirement grades classify NFRs into six categories: availability, performance/scalability, operability/maintainability, migratability, security, and system environment/ecology. Descriptions related to each category were labeled with the respective classification and used as training data. As shown in Table 4, the training dataset comprises a total of 933 instances, distributed across the six NFR categories.

For each experiment, a necessary number of instances was randomly extracted from this dataset. This approach ensured that the model could be fine-tuned effectively while utilizing a representative sample of the data for each specific experiment. Figure 3 shows the t-SNE plot of descriptions for each non-functional requirement grade after embedding them using the text-embedding-3-large model provided by OpenAI, which converts the text into 3072-dimensional vectors. Each color represents one of the six NFR categories. The clear clustering of some categories, like security, suggests that the classification task for these categories might be relatively straightforward for the model, likely resulting in higher accuracy, precision, and recall. Conversely, the dispersion observed in some categories, like the system environment/ecology categories, indicates potential challenges by enhancing context understanding. in achieving high classification performance. These categories might require more sophisticated models or additional context to improve classification accuracy. The t-SNE visualization serves as a validation tool, demonstrating that the embedding model can capture and represent the similarities and differences among the NFR descriptions to a

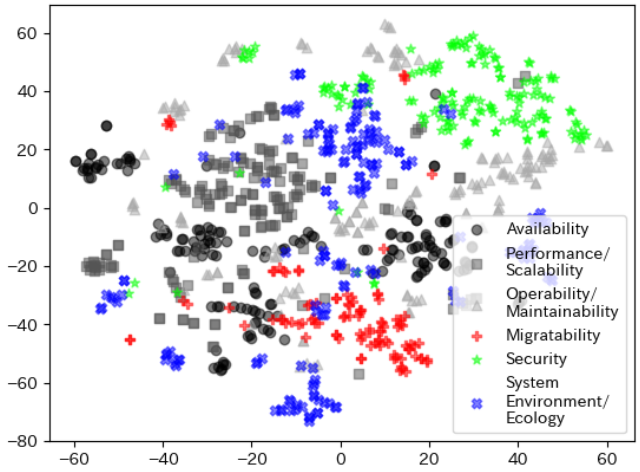

Figure 3 t-SNE Visualization of Embedded Training Data

significant extent. This visual validation supports the choice of using for fine-tuning and classification tasks.

In conclusion, the t-SNE plot effectively illustrates the clustering behavior of the NFR descriptions based on their embeddings. The distinct clusters for some categories and the overlapping regions for others provide valuable insights into the complexity of the classification task. These insights highlight the strengths and potential context understanding in using the current model for classifying non-functional requirements, guiding future improvements, and fine-tuning efforts.

5.2 Fine Tuning Effect

In order to comprehensively evaluate the progression of loss and its impact on accuracy, three different fine-tuning models were tested, each designed to address varying conditions in terms of dataset size and number of epochs, thus providing a robust analysis.

In finetuning 1, illustrated in Fig. 4-1, training was conducted over three epochs using 50 randomly selected training data samples. The loss progression in this scenario exhibits significant fluctuations in the initial stages but stabilizes in the later stages. Considering the small size of the training dataset, this early stabilization is presumed to lead to insufficient accuracy over broad data.

In finetuning 2, depicted in Fig. 4-2, the training dataset was increased to 250 randomly selected samples, and training was conducted over three epochs. Compared to the first scenario, the adaptation to a more diverse training dataset results in greater and more prolonged fluctuations. However, it is inferred that this leads to more advanced learning.

In finetuning 3, shown in Fig. 4-3, 250 training data samples were used, but the model was trained over nine epochs. In this case, the loss stabilizes within one epoch, suggesting that increasing the number of epochs does not necessarily aid in learning more from the data.

From these observations, it is evident that the size of the training data and the number of epochs have a significant impact on the learning process.

Figure 5 compares the classification accuracy of these three fine-tuned GPT-3 Turbo models and GPT-4 variants.

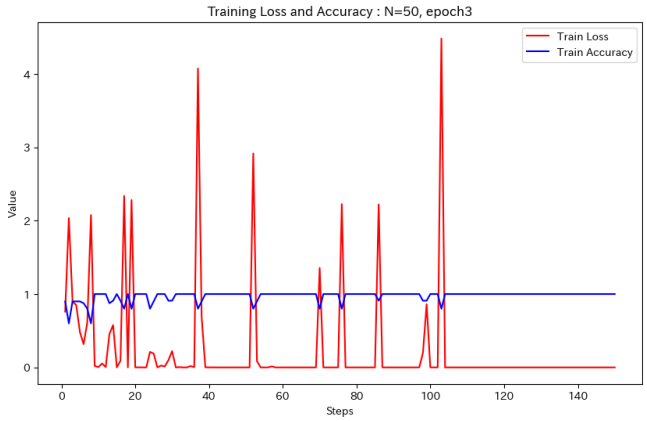

Figure 4 - 1 Training Loss and Accuracy over Epochs: N=50, Epoch=3 (red: train loss, blue: train Accuracy)

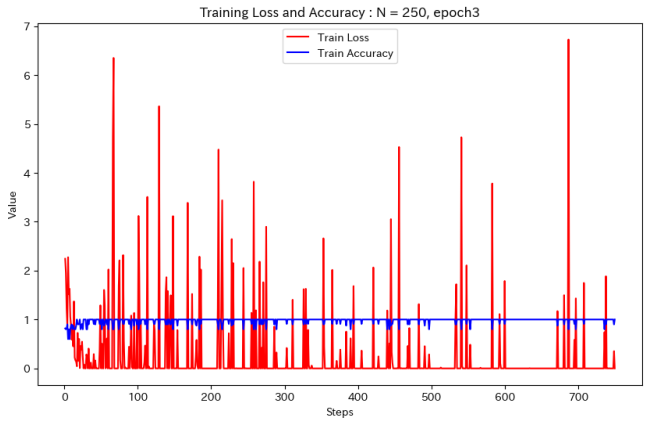

Figure 4 - 2 Training Loss and Accuracy over Epochs: N=250, Epoch=3 (red: train loss, blue: train Accuracy)

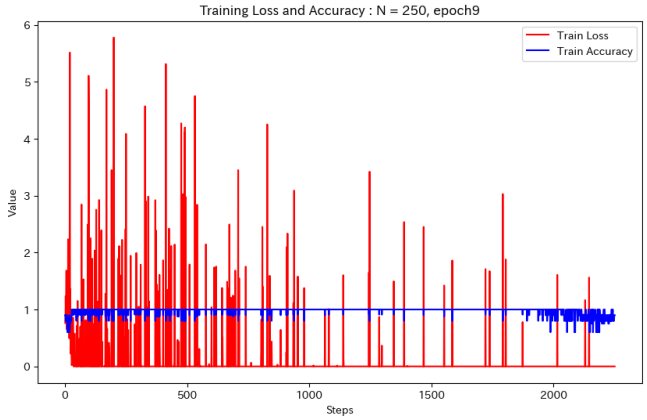

Figure 4 - 3 Training Loss and Accuracy over Epochs: N=250, Epoch=9 (red: train loss, blue: train Accuracy)

Among the metrics of Accuracy, Precision, Recall, and F1-score, we specifically chose the F1-score due to its balanced representation of both Precision and Recall. In both scenarios, where the training data consists of either 50 or 250 randomly selected samples, fine-tuning significantly improves the F1 score, reaching levels comparable to the accuracy achieved using Function Calling in GPT-4 variants. Increasing the training data from 50 to 250 samples results in a slight improvement in the F1 score. However, no additional improvements were observed with an increase in the number of epochs.

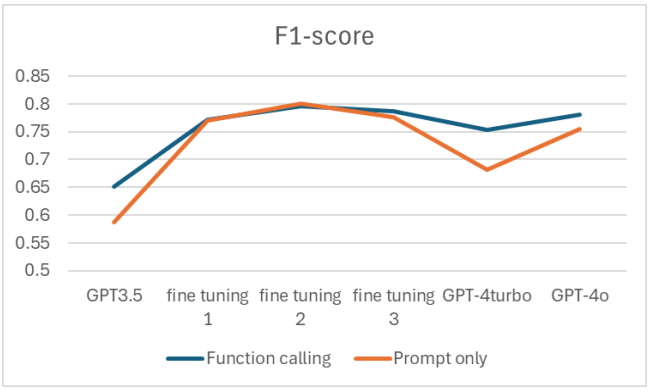

Figure 5 Comparison of F1-Scores for Function Calling and Prompt-Only Approaches Across Different Models and Fine-Tuning Stages

While using Function Calling without fine-tuning is effective, combining it with fine-tuning achieves higher classification accuracy. Furthermore, fine-tuning reduces the performance gap between using Function Calling and using only prompts. These findings provide valuable insights for selecting the optimal model design and training strategy for complex tasks such as non-functional requirements classification.

5.3 Improve Context Understanding

This experiment evaluated the effectiveness of the proposed sentence combination method by comparing it to analyses conducted without sentence concatenation. The classification accuracy results for each metric are shown in Table 5.

As illustrated in the table, the classification method using sentence combination (Combined Analysis) outperforms the no concatenation analysis across all evaluation metrics. Specifically, the accuracy improved.

Out of 414 experimental data points, 22 data points that were previously incorrect were corrected by using sentence combination, whereas 6 data points that were previously correct were misclassified. These results indicate that incorporating contextual information enhances classification accuracy, although it also introduces new misclassifications.

Basic no concatenation analysis methods tend to overlook contextual information as they consider only the target sentences for classification. In contrast, the method using sentence combination adds contextual information by combining preceding and/or succeeding sentences, allowing for more accurate classification. This approach is particularly effective for documents like requirement specifications, where non-functional requirements often span multiple sentences.

Table 5 Classification Accuracy Results Comparing the Proposed Sentence Combination Method and no Concatenation Analysis

	without Sentence Concatenation	Combined Analysis
Accuracy	0.792	0.831
Precision	0.785	0.839
Recall	0.779	0.808
F1-score	0.780	0.818

From these results, it can be confirmed that the classification method using sentence combination is effective, especially in the classification of non-functional requirements, by leveraging contextual information to improve classification accuracy. However, the increase in misclassifications suggests that further improvements and optimizations are necessary.

6 DISCUSSION

The primary aim of this study was to demonstrate that incorporating the Function Calling feature into ChatGPT's classification process can enhance the accuracy of NFR classification compared to relying solely on prompts. By adding Function Calling to the same prompt, the method achieves improved accuracy by reducing ambiguities and ensuring adherence to predefined classification criteria, even in a zero-shot setting.

This improvement, as indicated by accuracy increases across all models tested, is primarily attributed to the structured data retrieval and contextually appropriate categorization enabled by Function Calling. For example, GPT-4 Turbo demonstrated an improvement in F1 score, increasing from 0.681 to 0.753. Similarly, GPT-3.5 Turbo improved from 0.587 to 0.651, while GPT-4o achieved a gain from 0.754 to 0.780.

These results highlight that the integration of Function Calling consistently enhances classification accuracy across models by ensuring adherence to predefined classification boundaries. By defining enumerated categories, the model effectively avoids hallucinations, ensuring that the classifications adhere strictly to the intended boundaries. This capability highlights the model’s alignment with human reasoning in complex classification tasks.

Furthermore, the proposed method is practical as a primary classification tool, particularly when secondary human review is integrated into the process. This approach offers high efficiency and reliability. Additionally, considering that even human reviewers may have differing opinions on certain classification targets or that some items inherently span two categories, the practicality of this method becomes even more evident. However, in critical projects, robust secondary reviews are indispensable, and further research and development are necessary to enhance classification accuracy.

The supplementary experiments aimed to enhance classification accuracy by fine-tuning the model and incorporating contextual information, resulting in measurable improvements in precision and recall metrics.

Fine-tuning demonstrated that even minimal adjustments on small datasets could yield notable improvements in classification accuracy. For instance, the F1 score of GPT-3.5 Turbo improved significantly from 0.651 to 0.796, highlighting the effectiveness of fine-tuning in enhancing the model's performance.

Similarly, the use of sentence concatenation to incorporate contextual information from adjacent sentences enhanced the model's understanding, leading to more precise classifications. Specifically, this approach improved accuracy from 0.792 to 0.831, demonstrating the value of leveraging contextual information in classification tasks.

However, these techniques were positioned as complementary to the primary use of Function Calling.

In conclusion, Function Calling proved to be an essential feature, enabling consistent improvements in NFR classification accuracy over prompt-only methods. The supplementary methods, while beneficial, serve primarily to augment the foundational improvements achieved through Function Calling. Future research will focus on refining these methods and exploring their broader applications.

7 SUMMARY OF CONTRIBUTION AND CONCLUTIONS

This paper proposed and demonstrated the effectiveness of an automatic classification method for NFRs using Function Calling with large language models.

The Function Calling capabilities of GPT-4 Turbo and GPT-4o achieved excellent F1 scores of 0.753 and 0.780, respectively, matching or exceeding the performance of traditional task-specific models such as CNN or BERT on datasets similar to those used in this study for NFR classification tasks, even in a zero-shot learning state, demonstrating their high capability.

However, instances of misclassification were observed, particularly in scenarios involving ambiguous or overlapping categories, indicating the need for improvements in the model's contextual understanding abilities.

Two additional approaches, namely fine-tuning and sentence concatenation, were employed to enhance classification accuracy and address contextual ambiguities.

First, GPT-3.5 Turbo, with minimal fine-tuning, achieved results comparable to GPT-4 variants, emphasizing the importance and efficiency of fine-tuning. The significance of fine-tuning is evident, as it enhances the model's contextual understanding and classification accuracy.

Notably, the performance of the prompt-only approach also improves significantly with fine-tuning, as evidenced by an F1 score increase from 0.587 to 0.800 at maximum, ultimately narrowing the performance gap between the two methods.

Furthermore, experiments involving the concatenation of related sentences, by linking the sentences preceding and following the target for classification, confirmed the effectiveness of this approach in improving contextual understanding.

Future research should address these challenges by focusing on enhancing the model's contextual understanding

and creating targeted training datasets that emphasize the model's most challenging classification categories of NFRs.

ACKNOWLEDGEMENT

This research is being partially conducted as Grant-in-Aid for Scientific Research C (21K11826).

REFERENCES

[1] Japan Information Service Industry Association REBOK Planning WG, Requirements Engineering Body of Knowledge, Kindaikagaku-sha, pp. 1-200 (2011).
[2] Ministry of Economy, Trade, and Industry, Guideline for Improving the Reliability of Information Systems, 2nd Edition, pp. 1-50 (2009).
[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin, "Attention is all you need," Advances in Neural Information Processing Systems, Vol. 30, pp. 6000-6010 (2017).
[4] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL, Vol. 1, No. 1, pp. 4171-4186 (2019).
[5] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, pp. 1-10 (2018).
[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI, pp. 1-15 (2019).
[7] K. Mukaida, S. Fukui, T. Nagaoka, T. Kitagawa, S. Ogata, and K. Okano, "Efficient automatic classification of non-functional requirements in information systems using deep learning," IEICE Technical Report, Vol. 123, pp. 13-18 (2023).
[8] Information-Technology Promotion Agency, Japan, System Infrastructure Non-Functional Requirements Related Grade Table, Apr. 2013.
[9] T. Kinoshita, T. Omori, and J. Onishi, "Extraction and validation of non-functional requirements from Japanese requirements documents," IPSJ Technical Report (SE), Vol. 2021.15, pp. 1-6 (2021).
[10] R. K. Gnanasekaran, S. Chakraborty, J. Dehlinger, and L. Deng, "Using Recurrent Neural Networks for Classification of Natural Language-based Non-functional Requirements," Proc. of the 4th Workshop on Natural Language Processing for Requirements Engineering, CEUR Workshop, Vol. 2857, pp. 1-10 (2021).
[11] T. Kitagawa, T. Nagaoka, "Proposal and evaluation of an automatic classification method for requirement specifications using deep learning," IPSJ Transactions, Vol. 61, No. 4, pp. 842-852 (2020).
[12] Y. Zhu, Y. Wang, J. Qiang, and X. Wu, "Prompt-Learning for Short Text Classification," IEEE Trans. Knowledge and Data Engineering, Vol. 36, No. 10, pp. 5328–5339 (2024).
[13] J. Liu and L. Yang, "Knowledge-Enhanced Prompt Learning for Few-Shot Text Classification," Big Data and Cognitive Computing, Vol. 8, No. 4, Article 43, pp. 1-10 (2024).
[14] B. J. Gutierrez, N. McNeal, C. Washington, Y. Chen, L. Li, H. Sun, and Y. Su, "Thinking about GPT-3 In-Context Learning for Biomedical IE? Think Again," Findings of the Association for Computational Linguistics: EMNLP, pp. 4497-4512 (2022).
[15] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang, "Text classification via large language models," Findings of the Association for Computational Linguistics: EMNLP, pp. 8990-9005 (2023).
[16] S. Ibe, S. Kurata, T. Nagaoka, A. Furuhata, K. Goto, S. Fukui, and T. Kitagawa, "Automatic classification of requirement specifications using large language models," Proc. of the Software Engineering Symposium 2023, pp. 86-92 (2023).
[17] Y. Tan, D. Min, Y. Li, W. Li, N. Hu, Y. Chen, and G. Qi, "Can ChatGPT Replace Traditional KBQA Models? An In-Depth Analysis of the Question Answering Performance of the GPT LLM Family," Proc. of the International Semantic Web Conf., pp. 348-367 (2023).
[18] T. Brown, et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, Vol. 33, pp. 1877-1901 (2020).
[19] Q. Dong, S. Jiang, and W. Liu, "A survey for in-context learning," arXiv preprint arXiv:2301.00234, pp. 1-20 (2022).
[20] S. Min, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?," Proc. of the Conf. on EMNLP, pp. 11048-11064 (2022).
[21] Z. Wan, J. Xu, and M. Huang, "GPT-RE: In-context learning for relation extraction using large language models," arXiv preprint:2305.02105, pp. 1-10 (2023).

(Received: November 14, 2024)
(Accepted: August 12, 2025)

Kazuhiro Mukaida received his ME degree from Shinshu University in 2024 and is currently pursuing his doctoral studies at the same institution. His research focuses on natural language processing using large language models (LLMs) with the goal of enhancing the efficiency of requirements engineering processes.

Seiji Fukui received his M.S. in Natural Science from Okayama University in 2019. He is currently working for Toshiba Corporation. His research interests include software development environments and the application of AI technologies to software engineering.

Takeshi Nagaoka received his M.I. and Ph.D. degrees in Information Science and Technology from Osaka University in 2008 and 2011, respectively. He is currently working for Toshiba Corporation. His research interests include formal methods, software development environments, and the application of AI technologies to software engineering.

Takayuki Kitagawa received his M.E. degree in Management and Information Science from the Prefectural University of Hiroshima in 2008. He is currently working for Toshiba Corporation. His research interests include requirements engineering, software lifecycle process, and the application of AI technologies to software engineering.

Shinpei Ogata is an Associate Professor at Shinshu University, Japan. He received his B.E., M.E., and Ph.D. from Shibaura Institute of Technology in 2007, 2009, and 2012, respectively. He served as an Assistant Professor at Shinshu University from 2012 to 2020 and has been an Associate Professor there since 2020. He is a member of IEEE, ACM, IEICE, IPSJ, and JSSST. His current research interests include model-driven engineering for information system development.

Kozo Okano received his BE, ME, and PhD degrees in Information and Computer Sciences from Osaka University in 1990, 1992, and 1995, respectively. He was an Assistant Professor and an Associate Professor of Osaka University. In 2002 and 2003, he was a visiting researcher at the Department of Computer Science of the University of Kent in Canterbury, and a visiting lecturer at the School of Computer Science of the University of Birmingham, respectively. Since 2020, he has been a Professor at the Department of Electrical and Computer Engineering, Shinshu University. Since 2023, he has been the Director of Center for Data Science and Artificial Intelligence. His current research interests include formal methods for software and information system design and applying deep learning to Software Engineering.He is a member of IEEE, IEICE, and IPSJ.