LLM ContextBridge: A Hybrid Approach for Intent and Dialogue Understanding in IVSR

Changwoo Chun, Daniel Rim, Juhee Park
Hyundai Motor Company
Seoul, South Korea
{cwchun, drim, juheepark}@hyundai.com

Abstract

In-vehicle speech recognition (IVSR) systems are crucial components of modern automotive interfaces, enabling hands-free control and enhancing user safety. However, traditional IVSR systems often struggle with interpreting user intent accurately due to limitations in contextual understanding and ambiguity resolution, leading to user frustration. This paper introduces *LLM ContextBridge*, a novel hybrid architecture that integrates Pretrained Language Model-based intent classification with Large Language Models to enhance both command recognition and dialogue management. *LLM ContextBridge* serves as a seamless bridge between traditional natural language understanding techniques and LLMs, combining the precise intent recognition of conventional NLU with the contextual handling and ambiguity resolution capabilities of LLMs. This approach significantly improves recognition accuracy and user experience, particularly in complex, multi-turn dialogues. Experimental results show notable improvements in task success rates and user satisfaction, demonstrating that *LLM ContextBridge* can make IVSR systems more intuitive, responsive, and context-aware.

1 Introduction

In-Vehicle Speech Recognition (IVSR) systems play a vital role in modern vehicles by enabling hands-free control of the infotainment system, enhancing user safety. These systems are designed to handle single-turn commands, and often struggle with complex scenarios, such as multi-turn conversations, ambiguous utterances, and context-dependent inputs. As drivers need to concentrate on driving, it is very common for users to leave relevant details out in utterances.

Large Language Models (LLMs) offer a potential solution by improving dialogue management and resolving contextual ambiguities. However, fully integrating LLMs into IVSR systems can lead to challenges such as increased latency, computational costs, and inconsistent task performances. A full LLM-based approach is often inefficient in handling both simple and complex commands in production environments.

To address these challenges, we propose *LLM ContextBridge*, a hybrid architecture that combines the strength of a Pretrained Language Model (PLM)-based intent classification with the LLMs' contextual reasoning capabilities. *LLM ContextBridge* refines ambiguous or multi-turn utterances and ensures accurate intent recognition without requiring extensive retraining.

The key contributions of this work are:

- **Utterance Refinement:** *LLM ContextBridge* resolves ambiguities in both commands and conversations, enhancing intent classification without altering the existing natural language understanding (NLU) logic and dialogue management structure.
- **Contextual Multi-Turn Dialogue Handling:** *LLM ContextBridge* enables the system to manage complex dialogues while maintaining user context and intent.
- **Seamless Integration:** *LLM ContextBridge* incorporates LLMs into IVSR systems without requiring extensive fine-tuning, maintaining efficiency and performance.

2 Related Works

2.1 Existing Systems

IVSR systems are typically designed to handle single-turn commands by processing user inputs through intent classifiers and slot extractors (Lim et al., 2022). While these systems are effective for simple tasks, they struggle with multi-turn dialogues, where users' commands may omit critical information, or rely on context from previous interactions (Ferreira Cruz et al., 2020). For example, when a user asks, "What's the weather in Gangnam today?" followed by, "What about tomorrow?" the system fails to capture key terms like "Gangnam" and "weather" unless explicit mechanisms for retaining context are implemented (Hindle and Rooth, 1993).

In addition, IVSR systems face challenges in handling ambiguous utterances, such as "It's too noisy," which could refer to multiple aspects of in-car environment, including external noise or in-vehicle sound systems. Current approaches often rely on out-of-domain (OOD) detection (Jacobs et al., 2021; Papadopoulos et al., 2021; Wang and Mine, 2022; Shen et al., 2021) to identify unsupported or misclassified commands, but these solutions address only the detection aspect, rather than resolving the underlying ambiguity or context loss in multi-turn interactions.

2.2 LLM-based Approaches

Large Language Models (LLMs), trained on vast datasets and fine-tuned for task-specific applications (Wei et al., 2021), have shown promising results (He and Garner, 2023; Zhu et al., 2024) in enhancing the contextual understanding and dialogue management capabilities of IVSR systems. LLMs can handle more complex queries, and maintain context across multiple turns in a conversation, generating more natural responses (Brown et al., 2020; Zhao et al., 2023). The performance improvement is significant when given a few shots, and even better when applying a CoT (Wei et al., 2022) approach.

However, fully integrating LLMs into IVSR systems presents several challenges, such as increased latency and computational costs, making them less suitable for real-time production environments (Ma et al., 2023; Parikh et al., 2023). Moreover, while LLMs excel in dialogue generation, their performance in intent classification can be inconsistent, especially when dealing with a large number of intents. This makes LLMs less efficient for production-level IVSR systems that need to be fast and precise.

2.3 Hybrid Systems

To overcome the limitations of both traditional IVSR systems and LLM-based approaches, hybrid systems that combine rule-based methods with LLMs have been explored. Previous study (Rony et al., 2023) uses a retrieval-augmented generation model to answer user queries about vehicle features. This system employs an arbitration module that determines whether to use rule-based methods for simpler commands or LLMs for more complex questions. However, this approach can create a disjointed user experience, as the system may fail to handle transitions between complex and simple commands smoothly.

The orchestration required to combine two disparate systems introduces additional complexity. Bridging the gap between recognizing intents from utterances and carrying on a conversation is a major challenge.

3 Proposed Method

3.1 Overview of *LLM ContextBridge*

We propose a hybrid architecture, *LLM ContextBridge*, which combines the generative capabilities of LLMs with conventional NLU systems. This leverages LLMs to handle ambiguous, multi-turn dialogues that conventional NLU systems struggle with, while maintaining the overall structure and efficiency of rule-based and machine-learning-based NLU systems. Figure 5 illustrates the overall architecture, demonstrating how *LLM ContextBridge* integrates these systems to enhance intent classification and dialogue continuity in IVSR scenarios.

$$S(U,C) = \begin{cases} N_{\text{Rules}}(U) & \text{if } U \text{ is predefined} \\ N_{\text{PLM}}(L(U,C)) & \text{otherwise} \end{cases} \tag{1}$$

- $S(U,C)$: The IVSR system that processes user utterance U and context C.
- $N_{\text{Rules}}(U)$: Rule-NLU that handles predefined or patterned utterances U.
- $N_{\text{PLM}}(U))$: PLM-NLU that processes the free-form utterances.
- $L(U,C)$: The refined utterance generated by *LLM ContextBridge* using U and context C.

3.2 Conventional NLU Components

The conventional NLU system consists of two main components: rule-based NLU (Rule-NLU) and machine-learning-based NLU (PLM-NLU). Rule-NLU is responsible for processing well-defined, unambiguous commands such as “Navigate home” or “Make a call”. These commands follow predefined patterns that are straightforward to handle with a set of deterministic rules.

In contrast, PLM-NLU handles more flexible, free-form utterances that cannot be fully predefined. PLM-NLU excels when it is trained on large

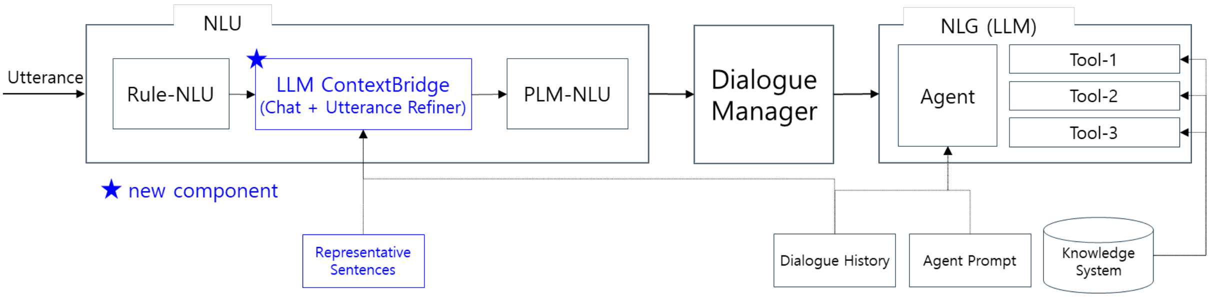


Figure 1: The Overall Architecture of *LLM ContextBridge*

datasets with well-defined intents and proper nouns, such as points of interest (POIs) or song titles. This allows the system to perform intent classification and slot extraction tasks with a high precision. However, when faced with utterances outside the predefined domain (e.g., "the window is broken" when only "open window" and "close window" are recognized), the system encounters out-of-domain issues, which can hinder intent recognition.

For further details on the baseline system's architecture, please refer to Section A in the appendix.

3.3 The Role of *LLM ContextBridge*

LLM ContextBridge addresses the limitations of the conventional NLU by introducing LLMs to handle complex, multi-turn dialogues. It processes utterances that are ambiguous, context-dependent, or contain ellipses, refining them to ensure the user's intent is fully captured. This refinement occurs before the utterance is passed to the PLM-NLU for final processing.

LLM ContextBridge uses both the user's current utterance U and the preceding dialogue context C to refine U. By restoring the omitted context, modifying utterances to align with predefined forms, and clarifying ambiguous statements through follow-up questions, *LLM ContextBridge* ensures that the system can handle more complex dialogues with greater accuracy.

The refined utterance is then passed to the PLM-NLU for further processing, as represented by the following equation:

$$L(U, C) \rightarrow N_{\text{PLM}}(L(U, C)) \qquad (2)$$

3.4 Multi-turn Dialogue Handling

LLM ContextBridge is specifically designed to handle the challenges of multi-turn dialogues, where the meaning of an utterance evolves based on prior interactions. The following tasks are performed through prompt strategies applied to the LLM. For detailed prompt configurations, please refer to the appendix C.

The multi-turn dialogue handling can be categorized into four main cases:

Handling Specification Utterances For utterances explicitly defined in the specification, *LLM ContextBridge* passes them directly to the PLM-NLU for processing. For similar but not identical utterances, *LLM ContextBridge* refines them to match predefined forms. For instance, "Let's go to Lotte Tower" becomes "Navigate to Lotte Tower" to ensure consistent classification.

Handling Ambiguous Utterances *LLM ContextBridge* clarifies ambiguous utterances with follow-up questions. If the user says, "It's too noisy," the system might ask, "Do you want to lower the volume?" Once confirmed, the system refines the utterance to "Turn down the volume."

Restoring Omitted Information The system restores omitted details based on context. For example, "Let's go there" could be refined to "Let's go to Starbucks," and "Only the driver's seat" to "Turn on the air conditioning for only the driver's seat."

Handling External Knowledge For queries requiring external knowledge (e.g., real-time traffic), *LLM ContextBridge* identifies the appropriate API, retrieves the needed data, and uses it to generate a response.

3.5 Integration of *LLM ContextBridge* with NLU

For seamless integration, *LLM ContextBridge* sits between the Rule-NLU and PLM-NLU components. Rule-NLU handles simple, well-defined utterances, while *LLM ContextBridge* processes more complex or ambiguous utterances based on context. If the system determines that the utterance cannot be handled by the conventional NLU, *LLM ContextBridge* takes over, ensuring that user intent is accurately interpreted, and the dialogue remains natural.

By bridging the gap between conventional NLU and LLMs, *LLM ContextBridge* creates a hybrid system that retains the precision of traditional systems while adding the flexibility and conversational capabilities of LLMs.

3.6 Advantages of *LLM ContextBridge*

The integration of *LLM ContextBridge* offers several advantages:

- **Seamless Refinement:** Refines ambiguous or incomplete utterances, ensuring accurate intent capture.
- **Contextual Awareness:** Leverages dialogue context to maintain coherence across multi-turn dialogues.
- **Hybrid Efficiency:** Balances Rule-NLU precision with LLM flexibility, processing simple utterances efficiently while handling complex ones effectively.
- **Out-of-Domain Handling:** Transforms unsupported utterances into processable forms for PLM-NLU.

4 Experiments

4.1 Data & Models Specifications

To evaluate our proposed method, we used three datasets based on real user logs and compared the two systems. Both systems use the same Rule-NLU and PLM-NLU (fine-tuned from ELECTRA (Clark et al., 2019)). For the proposed method, *LLM ContextBridge* integrates GPT-4o[1] as the LLM component, chosen for its proven ability to handle Korean. We also conducted comparative experiments using the open LLM, LLaMA-3.1[2] (et. al., 2024), which showed lower performance in handling Korean-language tasks.

IVSR Evaluation dataset: Functions-Set The Functions-set consists of 13,138 utterances, covering 282 intents across 12 domains. Each domain has 10 to 30 intents, reflecting real-world variability. Utterances were annotated based on user logs, with the domain distribution shown in Figure 2. Major domains include [Infotainment system control] for media and volume commands, and [Vehicle Control] for tasks like windows, sunroof, and climate control. Meanwhile, the [Agent] domain represents dialogue-driven requests like general conversations.

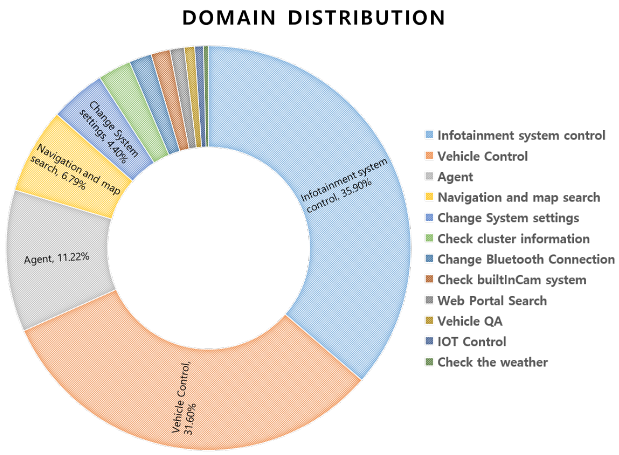


Figure 2: Domain distribution of the utterances in the Functions-set

Conversational Evaluation: Dialogue-Set To assess the system's performance on multi-turn dialogues, we curated a new dataset, derived from actual user logs and extended through simulation. The Dialogue-set consists of both single-turn and multi-turn conversations. First, we gathered a single-turn evaluation set containing function-execution commands and question-answer pairs. Multi-turn dialogues were then generated using a simulated interaction between user and system agents, both modeled by GPT-4o, as illustrated in Figure 3. There were 5,501 single-turn conversations and 1,697 multi-turn conversations, with the average number of utterances in a multi-turn conversation being 4.37.

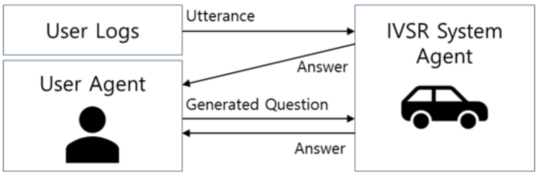


Figure 3: Multi-turn dialogue generation process

4.2 Evaluation Methods and Criteria

The systems were evaluated based on:

- **User Request Handling Accuracy**: Accuracy of matching actions or responses to the user's utterance.
- **Response Appropriateness**: How correctly the system make relevant responses in the Dialogue-set.

[1] https://openai.com/gpt-4o-contributions/

[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

- **Naturalness of Dialogue**: How naturally the system maintains context in dialogues.

	Quantitative Eval	Qualitative Eval
Method	G-Eval	Human evaluation
Criteria	1) User request handling accuracy 2) Response appropriateness 3) Naturalness of dialogue	1) User request handling accuracy 2) Response appropriateness 3) Naturalness of dialogue

Table 1: Evaluation methods and criteria

Quantitative evaluation was performed using the GPT-4o model (Zheng et al. 2024), while qualitative evaluation involved three human evaluators rating 60 dialogues based on the same criteria.

4.3 Performance Evaluation Results and Analysis

Intent Classification Tasks We compared *LLM ContextBridge* system with the conventional NLU (Baseline) and LLM-only systems using the Functions-set. Table 2 shows the intent classification accuracy, and Figure 4 provides domain-specific F1-scores.

Method	LLM Used	Acc.
Baseline	N/A	0.896
Proposed	GPT-4o	**0.917**
	LLaMA-3.1-8B-instruct	0.782
LLM-only	GPT-4o	0.636
	LLaMA-3.1-8B-instruct	0.497

Table 2: Intent Classification Accuracy Across Different Methods on the Functions-set

The baseline system, combining Rule-NLU and PLM-NLU, achieved an accuracy of 0.896. Despite the complexity of the test set, which includes ambiguous or context-dependent utterances, this demonstrates the robustness of the PLM-NLU model trained on large-scale data.

In the proposed method, *LLM ContextBridge* refines user utterances before PLM-NLU processes them. Using GPT-4o, the proposed system achieved 0.917 accuracy, a 2.1% improvement over the baseline. This highlights the benefit of LLM's generative capabilities in refining complex utterances. However, using LLaMA-3.1-instruct, accuracy dropped to 0.782, primarily due to its limited proficiency in handling Korean-language tasks.

The LLM-only approach, without the conventional NLU, performed significantly worse. GPT-4o reached the accuracy of 0.636, and LLaMA-3.1-instruct only achieved 0.514. These results illustrate the difficulty LLMs face with large amounts of multi-class classification without conventional NLU support.

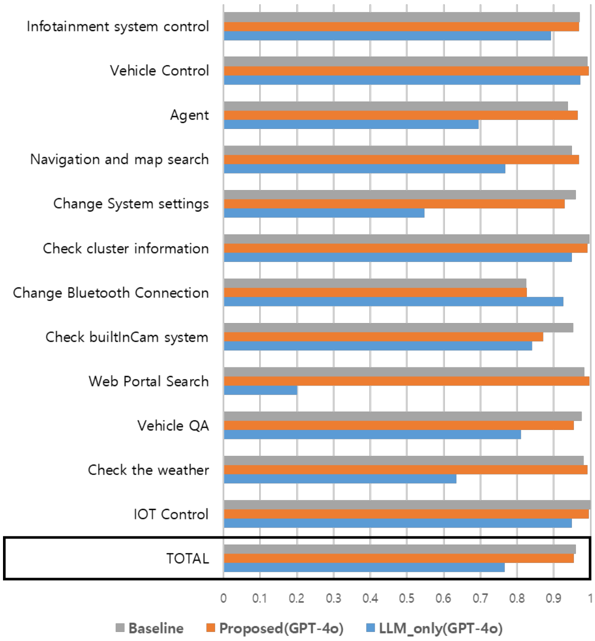


Figure 4: Domain-specific F1-Scores

In terms of domain-specific performance (Figure 4), the proposed system (0.955) showed similar performance to the baseline (0.960) in most domains. In "Agent" domain, where conversational performance is important, there was a slight improvement, where the F1 score increased from 0.939 of the baseline to 0.965. In contrast, the LLM-only system underperformed in all domains, particularly in complex tasks like "Web Portal Search" and "IOT Control." For detailed performance metrics across each domain for these systems, see Appendix 7, 8, and 9.

We explored the nuanced differences in system responses to variations in window-related utterances across three methodologies: Baseline, Proposed, and LLM-only. The Baseline system, due to a design bias towards prioritizing navigation functions within vehicle infotainment systems, interprets the simple utterance "window" as a prompt for a point-of-interest search, assuming it to refer to a location involving the word "window." In contrast, the LLM-only method, unguided by specific biases that could be feasibly applied to millions of POIs, defaults to treating the command as "open window," showing a misunderstanding

Command	Baseline	Proposed	LLM-only
Window	Search POI	Search POI	Open Window
Open window	Open Window	Open Window	Open Window
Can you open the window	Open Window	Open Window	Chat
Does it work to open the window	Open Window	Open Window	Chat
Should I open or close the window	Open Window	Chat	Chat
A way to open a window	How Open	How Open	Chat
Tell me how to open a window	How Open	How Open	How Open
Window opening speed	Open Window	Chat	Chat
Prevent the window opening	Close Window	Lock Window	Close Window
How to lock the window	How Lock	How Lock	How Lock

Table 3: Comparison of system responses to nuanced variations in window-related commands across different methods.

of context. Interestingly, while the Baseline approach misclassifies less explicit utterances like "Should I open or close the window?" as a command to "open window". The systems employing LLMs can navigate these ambiguities more adeptly, categorizing them as "Chat" and prompting a conversational interaction that asks for clarification, such as "Would you like me to open the window, or should I keep it closed?" This distinction underscores the LLM-based methods' superior ability to engage in context-sensitive dialogue. Although the LLM-based methods adeptly handle ambiguous utterances by prompting conversational interactions for clarification, they cannot be utilized exclusively, due to critical limitations. The LLM-only approaches often struggle with utterances that mimic the system's inherent functionalities, as they lack the design biases specifically tailored to interpret the system's native commands. Consequently, this can lead to a system either misinterpreting metaphorical requests or providing inaccurate explanations generated by the LLMs rather than precise, system-designed responses.

Overall, these results demonstrate that *LLM ContextBridge* successfully balances the strengths of both conventional NLU and LLMs, improving performance in handling complex dialogues and ambiguous utterances across domains.

Conversation Tasks We compared the performance of single-turn and multi-turn dialogues using the Dialogue-Set. Table 4 presents the evaluation results for both the baseline and proposed systems.

Dialogue Dataset	Baseline	Proposed
Single-turn	0.773	**0.892**
Multi-turn	0.152	**0.601**

Table 4: Evaluation results based on the GPT-4o model.

The proposed method exhibited substantial improvements in both single-turn and multi-turn dialogues. For single-turn dialogues, the proposed system achieved an accuracy of 0.892, reflecting a 12% improvement over the baseline (0.773). This shows that our approach effectively improves the interpretation of isolated commands, favoring a more nuanced understanding than the baseline to create appropriate responses to utterances.

The impact of *LLM ContextBridge* becomes even more pronounced in multi-turn dialogues. The proposed system reached an accuracy of 0.6005, a dramatic 45% improvement over the baseline's 0.1520. This highlights the system's ability to manage complex, context-dependent interactions, which were difficult for the conventional systems.

To further validate these findings, a qualitative evaluation was conducted through human assessments, as shown in Table 5.

Evaluator	Baseline	Proposed
Evaluator 1	0.15	**0.783**
Evaluator 2	0.083	**0.717**
Evaluator 3	0.2	**0.833**

Table 5: Qualitative evaluation results on multi-turn dialogue – Human evaluation.

The human evaluation results underscore the effectiveness of the proposed system in multi-turn dialogue scenarios. Despite some variations among the evaluators, the qualitative scores consistently indicate a significant improvement in performance with the proposed *LLM ContextBridge* system.

Overall, *LLM ContextBridge* not only improves the handling of single-turn commands, but also significantly enhances the performance of multi-turn dialogues by incorporating LLM-based utterance refinement. These results emphasize the contributions of a hybrid approach, particularly in the context of managing dialogues that require maintaining of context, and resolving ambiguity over multiple interactions.

4.4 Processing speed and Efficiency

While *LLM ContextBridge* demonstrated improved dialogue performance, it exhibited different response characteristics across test sets. As shown in Table 6, for the functions-set, the proposed system introduced an additional delay of up to 600ms, with the baseline system is faster than the proposed approach. This slower processing time is attributed to the computational overhead of LLM-based utterance refinement. However, in the dialogue-set, the proposed system showed a clear advantage, with faster response times by 300-500ms per turn in both single-turn and multi-turn dialogues.

Despite the increased latency in the functions-set, the response times for all scenarios remain well within the 3-second production-level timeout requirement for IVSR systems, ensuring that the proposed system maintains acceptable responsiveness for real-world applications.

Evaluation data	Baseline	Proposed
Functions-set	**0.272**	0.851
Dialogue: Single-turn	1.354	**1.052**
Dialogue: Multi-turn	2.136	**1.652**

Table 6: Comparison of processing speed (unit: sec)

5 Conclusion

In this paper, we presented *LLM ContextBridge*, a hybrid architecture that integrates Pretrained Language Models (PLMs) with Large Language Models (LLMs) to enhance intent classification and dialogue management in In-Vehicle Speech Recognition (IVSR) systems. By bridging traditional natural language understanding (NLU) with LLMs, *LLM ContextBridge* effectively addresses key challenges in handling complex, multi-turn dialogues, and resolving ambiguous commands.

Our experiments showed significant improvements, with a 12% increase in single-turn accuracy and a 45% improvement in multi-turn dialogues compared to the baseline system. These gains highlight *ContextBridge*'s ability to refine ambiguous utterances and maintain dialogue context, creating a more intuitive and responsive user experience. The hybrid approach also integrates seamlessly into existing IVSR systems without the need for extensive retraining, making it suitable for real-world applications. We believe *LLM ContextBridge* marks an important step forward for IVSR systems, improving their ability to handle context-dependent interactions and making them more user-friendly. The integration of LLMs with conventional NLU systems offers a path toward more intelligent, adaptable, and accessible automotive interfaces, enhancing both user satisfaction and safety.

6 Limitations

While *LLM ContextBridge* offers considerable advancements in improving intent classification and dialogue management within IVSR systems, several limitations remain that highlight areas for future improvement.

LLM Selection and Language Generalization: The choice of LLM plays a critical role in system performance. While GPT-4o demonstrated strong capabilities in handling Korean-language tasks, models like LLaMA-3.1-instruct showed lower performance in this area. This suggests that the effectiveness of *LLM ContextBridge* may vary significantly depending on the LLM's language understanding and adaptability. Evaluating other LLMs across different languages and further optimizing their integration into the system remains an important future direction.

Processing Speed and Efficiency: Incorporating an LLM introduces additional processing layers, which can impact real-time performance, particularly in time-sensitive in-vehicle environments. The observed latency—up to 600ms in some tasks—indicates a need for further optimization. Although the hybrid system improves contextual understanding, ensuring prompt responses for simpler commands without unnecessary LLM intervention is crucial for maintaining efficient processing.

Dependency on Predefined Utterances: *LLM ContextBridge* relies on predefined utterances and specification documents, particularly in handling structured commands. This reliance can limit the system's ability to manage entirely novel or unstructured utterances that deviate from established patterns. Enhancing the system's flexibility in addressing unforeseen commands is an ongoing challenge.

Human Evaluation and User Variability: Although qualitative evaluations indicated improvements in multi-turn dialogues, the inherent subjectivity of human assessments can lead to inconsistencies in results. The relatively small sample size in our evaluations may also cause variability depending on which specific samples were used, making it difficult to generalize the findings. As IVSR systems continue to evolve, it will be essential to address the diverse needs of both experienced and first-time users. This will require continuous refinement of system functionality and user interaction to ensure that the system performs effectively across a wide range of user experiences and expectations.

Baselines and Dataset: As our work is to demonstrate feasibility of our approach on a production level system, we limited our experiments to three systems: our current in-production system, a LLM-only system, and our proposed system. Furthermore, our work is focused on demonstrating the *improvements* that can be made by seamless incorporation of LLMs. Our method is adaptable to various systems, and can be utilized to improve other existing production systems. Lastly, as most of our experiments are performed on real user data, we are unable to provide more details regarding the dataset.

In conclusion, while *LLM ContextBridge* demonstrates significant potential for enhancing IVSR systems, addressing these limitations will be critical to ensuring its scalability and effectiveness in broader contexts.

Acknowledgments

This project has filed a patent with the number KR-1020240117134.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Abhimanyu Dubey et. al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

André Ferreira Cruz, Gil Rocha, and Henrique Lopes Cardoso. 2020. Coreference resolution: toward end-to-end and cross-lingual systems. *Information*, 11(2):74.

Mutian He and Philip N Garner. 2023. Can chatgpt detect intent? evaluating large language models for spoken language understanding. *arXiv preprint arXiv:2305.13512*.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120.

Pieter Floris Jacobs, Gideon Maillette de Buy Wenniger, Marco Wiering, and Lambert Schomaker. 2021. Active learning for reducing labeling effort in text classification tasks. In *Benelux Conference on Artificial Intelligence*, pages 3–29. Springer.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Jungwoo Lim, Suhyune Son, Songeun Lee, Changwoo Chun, Sungsoo Park, Yuna Hur, and Heuiseok Lim. 2022. Intent classification and slot filling model for in-vehicle services in korean. *Applied Sciences*, 12(23):12438.

Yubo Ma, Yixin Cao, Yong Hong, and Aixin Sun. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10572–10601, Singapore. Association for Computational Linguistics.

Aristotelis-Angelos Papadopoulos, Mohammad Reza Rajati, Nazim Shaikh, and Jiamian Wang. 2021. Outlier exposure with confidence control for out-of-distribution detection. *Neurocomputing*, 441:138–150.

Soham Parikh, Mitul Tiwari, Prashil Tumbade, and Quaizar Vohra. 2023. Exploring zero and few-shot techniques for intent classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 744–751, Toronto, Canada. Association for Computational Linguistics.

Md Rashad Al Hasan Rony, Christian Suess, Sinchana Ramakanth Bhat, Viju Sudhi, Julia Schneider, Maximilian Vogel, Roman Teucher, Ken Friedl, and Soumya Sahoo. 2023. CarExpert: Leveraging large language models for in-car conversational question answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 586–604, Singapore. Association for Computational Linguistics.

Yilin Shen, Yen-Chang Hsu, Avik Ray, and Hongxia Jin. 2021. Enhancing the generalization for intent classification and out-of-domain detection in slu. *arXiv preprint arXiv:2106.14464*.

Bo Wang and Tsunenori Mine. 2022. Practical and efficient out-of-domain detection with adversarial learning. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 853–862.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Zhihong Zhu, Xuxin Cheng, Hao An, Zhichang Wang, Dongsheng Chen, and Zhiqi Huang. 2024. Zero-shot spoken language understanding via large language models: A preliminary study. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17877–17883.

A Baseline System Architecture

The baseline system used in our experiments consists of three main components: **NLU (Natural Language Understanding)**, **DM (Dialogue Management)**, and **NLG (Natural Language Generation)**. These components are structured as follows:

A.1 NLU (Natural Language Understanding)

The NLU component is responsible for intent classification in single-turn utterances. It is divided into two subcomponents:

- **Rule-NLU:** This subcomponent processes predefined, well-structured utterances. It uses rule-based methods to classify intents for commands such as "Navigate home" or "Make a call." Rule-NLU is designed to handle only the top 5% of the most frequently used utterances.
- **PLM-NLU:** This subcomponent handles more flexible, free-form utterances that cannot be predefined by rules. It is fine-tuned from a Pretrained Language Model (PLM) and is responsible for identifying intents in utterances such as "How long does it take to get to the nearest Starbucks?" The PLM-NLU model is based on a bi-directional transformer text encoder architecture, specifically the ELECTRA

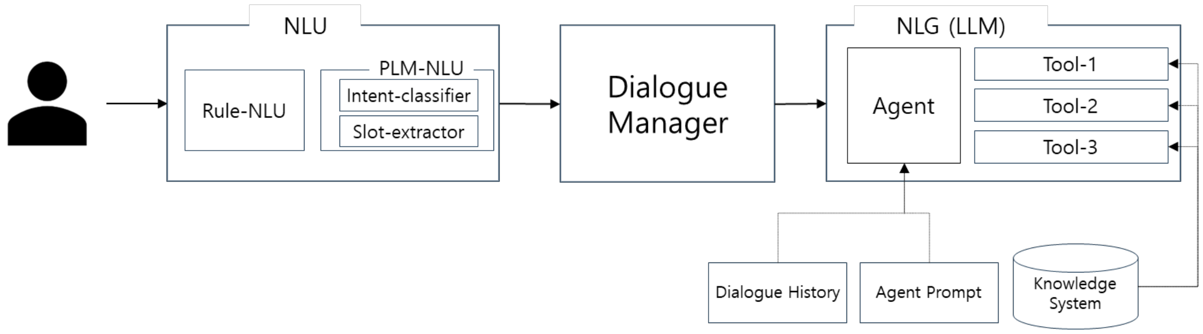


Figure 5: The Overall Architecture of *IVRS* system (without ASR & TTS)

base model, by training an intent classifier and a slot extractor, respectively. The model's key parameters are as follows:

- Architectures: `ElectraForSequenceClassification`
- Embedding Size: 768
- HiddenActivation: `gelu`
- HiddenSize: 768
- MaximumPositionEmbeddings: 512
- ModelType: `electra`
- NumberofAttentionHeads: 12
- NumberofHiddenLayers: 12
- PositionEmbeddingType: `absolute`
- ProblemType: `single_label_classification`
- SummaryActivation: `gelu`
- VocabularySize: 32,200

A.2 DM (Dialogue Management)

The DM component manages the dialogue flow. While it handles single-turn utterances, it is also designed to manage multi-turn interactions in specific scenarios where follow-up utterances and responses are required. The DM processes the user's input, tracks the conversation state, and selects the appropriate response templates to maintain the flow of dialogue.

A.3 NLG (Natural Language Generation)

The NLG component generates responses based on the templates processed by the DM. If the NLU or DM component cannot fully handle the request, or if an out-of-spec utterance is encountered, it uses an LLM to manage the question-answering task. The LLM complements the system by generating responses for out-of-domain queries or ambiguous utterances not predefined in the template library.

For queries requiring external resource access, such as retrieving Point of Interest (POI) information or accessing vehicle maintenance manual systems, the LLM Agent identifies the appropriate tool and makes API calls with relevant parameters to retrieve external knowledge. Once the results are fetched, the system generates responses using Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) in order to ensure accurate and contextually relevant answers.

A.4 Exclusion of ASR and TTS

In this study, we excluded the Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) components from the experiments. The focus was on the interaction between the NLU, DM, and NLG components, and the performance of these modules was evaluated independently of speech input and output systems.

B Detailed Experimental Setup

In our experimental setup, we used both the conventional NLU-based system and a hybrid architecture incorporating Pretrained Language Models (PLMs) and Large Language Models (LLMs) for intent classification and dialogue handling. Below, we detail the hardware, software environments, and models used for our experiments.

B.1 Hardware and Software Environment

All experiments were conducted using an NVIDIA GPU server with 8 Tesla A100 GPUs, each with 40GB of memory. The operating system was Ubuntu 20.04 LTS, and Python 3.8 was used as the programming language.

The experiments were run in a Docker container environment using the following setup:

- **CUDA version:** 11.3
- **Python libraries:** Transformers 4.30.2, Torch 1.10.0, and FastAPI 0.78

B.2 LLM Usage Parameters

For generating responses in tasks such as dialogue generation and question answering, we used a variety of models including GPT-4o, GPT-4o-mini, CCVR-Chat, and LLaMA-3.1-8B-Instruct. Below are the detailed parameters used for controlling the behavior of the LLM during inference:

- **Model Version:**
  - **GPT-4o:** gpt-4o-2024-05-13
  - **LLaMA-3.1-8B-Instruct:** llama-3.1-8B-instruct
- **Temperature:** 0.3,
- **Max Tokens:** 512 tokens,
- **Top-p:** 0.95,
- **Frequency Penalty:** 0 (default),
- **Presence Penalty:** 0 (default),

These parameters were fine-tuned to balance response quality, coherence, and efficiency, ensuring that the generated outputs were relevant and contextually appropriate for the IVSR system.

C LLM ContextBridge Prompt

LLM ContextBridge Prompt provides a structured approach to interacting with a In-Vehicle Speech Recognition (IVRS) system. It includes four main components.

C.1 Situation Description

> The following are representative commands for all functions supported by the Hyundai Motor Company's vehicle voice recognition system. The voice recognition system can recognize the driver's speech and execute the function or respond. The parts marked with "< >" in the supported representative commands are proper nouns, and the other parts represent semantic expressions. Various words can be included in the proper noun part.

C.2 Specification: Representative Commands

The system utilizes a set of abstracted representative commands to efficiently manage the range of functions it can understand and execute, such as checking the weather, controlling media playback, managing vehicle settings, and making calls. These representative commands are derived from a summary of the vehicle voice recognition specification, ensuring full coverage of In-Vehicle Speech Recognition System (IVRS) capabilities.

In *LLM ContextBridge*, we use abstracted representative commands to optimize the use of the LLM's input prompt length (max_token_length). This approach ensures the LLM can capture the full range of functionalities while minimizing input complexity. Traditional methods often rely on intent-example pairings provided in a few-shot manner, leading to a significantly longer prompt. However, the proposed method reduces the overall prompt length by using only abstracted representative commands without explicitly including intents, making the system more efficient in handling the available prompt space.

> A representative command is a representative sentence of all functions supported by the voice recognition system. Even if the customer's speech is somewhat different from this, if a different predicate with a similar meaning is used, and a minor difference in particle or change in ending is considered to have the same meaning as the representative command.

```
Tell me the weather in <REGION>
Tell me the weather on <DATE>
<BroadcastingStationName>
AM <Frequency>
FM <Frequency>
Call <CallTarget>
Send message <content> to <CallTarget>
Turn off <APP>
Run <APP>
Turn on <APP>
Turn on <AirConditioner>
Turn off <AirConditioner>
Raise the temperature of <AirConditioner>
Lower the temperature of <AirConditioner>
...
Directions <POI>
Find <POI>
Search for <POI>
Let's go to <POI>
Turn down the volume of <APP>
Turn up the volume of <APP>
Set the volume of <APP> to <NUMBER>
Mute <APP>
Unmute <APP>
Temperature Lower <NUMBER>
Raise <temperature>
Set temperature to <NUMBER>
Play <ARTIST>
Play <SongTitle>
Play <SongTitle> sung by <ARTIST>
Open <SWITCH>
Close <SWITCH>
Raise <SWITCH>
Put <SWITCH> down
```

C.3 Few-shot Examples

Examples demonstrate how the system processes different types of user input, from simple questions to requests for specific actions. These examples help the system handle both direct commands and interactive interactions without errors.

```
User: Who made you?
Assistant: {'function':'CHAT', 'question':'Who made you', 'answer':'I am an AI assistant made by Hyundai Motor Company.'}
User: Is there a Starbucks near Gangnam Station?
Assistant: {'function':'NLU', 'question':'Find Starbucks near Gangnam Station', 'answer':'I will find a Starbucks near Gangnam Station.'}
User: Tell me about the Hyundai Motor company's electric vehicle lineup
Assistant: {'function':'AI', 'question':'Tell me more
```

> about Hyundai Motor Company’s electric car lineup’, ’answer’:None}
> User: Tell me more
> Assistant: {’function’:’AI’, ’question’:’Tell me more about Hyundai Motor Company’s electric car lineup’, ’answer’:None}

C.4 Task Instructions

The instructions provide a structured approach for processing driver utterances within IVSR systems. The prompt details how to handle different types of inputs, ensuring the system generates contextually appropriate responses in JSON format.

> The assistant should return the result of processing the driver’s utterance as a JSON-formatted string. The response JSON entity contains three keys: `response = {’function’, ’question’, ’answer’}`.
> Process the driver’s utterance according to the following rules:
>
> 1. If the driver’s utterance has the same meaning as a representative command and corresponds to the same function, respond with a message confirming that the function is being executed. Insert this message as the value for both `’function’` and `’answer’`.
> 2. If there are multiple representative commands that are semantically similar but not identical to the recognized utterance, ask the driver for clarification. In this case, set `’function’` to `’CHAT’` and insert the assistant’s follow-up question as the value of `’answer’`.
>
>   If the driver responds with `’yes’` or another confirmation, update `’function’` to `’NLU’`, set `’question’` to the representative command inferred from the conversation, and provide a message confirming that the function is being executed in `’answer’`, based on the context.
> 3. If the utterance refers to a function that is not supported by the vehicle or voice recognition system, set `’function’` to `’CHAT’` and `’answer’` to a message stating, `"This is not a supported function in this vehicle."`
> 4. For utterances unrelated to the vehicle’s functions, provide a friendly and helpful response. Set `’function’` to `’CHAT’`, and insert the assistant’s response as the value of `’answer’`.
> 5. For queries requiring external information, such as searching for the latest information, external knowledge, or vehicle manual data, set `’function’` to `’AI’` and `’answer’` to `None`.
> 6. If `’question’` requires rephrasing to match a representative command (as in Rule 1), replace only the predicate part of the driver’s original utterance with the representative command, while preserving any proper nouns. In all other cases, use the driver’s original input in the `’question’` field.
> 7. If the driver’s current utterance omits details that are clear from the context, restore the omitted content in both `’question’` and `’answer’`.

D Example Use Cases

The case study below demonstrates the practical application of our system. It highlights how LLM ContextBridge has been effectively integrated into a real-world IVSR system, seamlessly integrating the existing NLU system with the generative capabilities of LLMs.

> User: It’s getting a bit chilly.
> Assistant: {’function’:’CHAT’, ’answer’:’It sure has gotten cold. Wear something warm! Would you like to increase temperature setting?’}
> User: Yes.
> Assistant: {’function’:’NLU’, ’answer’:’Okay, increasing the temperature setting.’}
> User: Good. Are there any good burger places nearby?
> Assistant: {’function’:’CHAT’, ’answer’:’The McDonald’s Songpa Jamsil DT branch is nearby.’}
> User: Sure, please tell me the route.
> Assistant: {’function’:’NLU’, ’answer’:’Navigating to the McDonald’s Songpa Jamsil DT branch.’}

> User: I smell something strange.
> Assistant: {’function’:’CHAT’, ’answer’:’If there is a strange smell inside the vehicle, please check the ventilation system. Would you like me to turn on the air cleaner mode?’}
> User: I do not know where the smell is from.
> Assistant: {’function’:’CHAT’, ’answer’:’It’s difficult to determine the exact cause of the smell inside the vehicle. However, turning on the air cleaner mode or opening the windows for ventilation might help. Would you like me to turn on the air cleaner mode?’}
> User: Yes.
> Assistant: {’function’:’NLU’, ’answer’:’Turning on the air cleaner mode.’}
> User: Adjust the airflow direction as well.
> Assistant: {’function’:’CHAT’, ’answer’:’How would you like to adjust the airflow? You can choose from upper airflow, lower airflow, or a combination of both.’}
> User: Towards my face.
> Assistant: {’function’:’NLU’, ’answer’:’Adjusting the airflow upwards, toward your face.’}

E Detailed Performance Analysis Across Domains

In this section, we provide a comprehensive analysis of the performance metrics for three distinct approaches to handling domain-specific tasks within our IVSR system. The following tables delineate the precision, recall, and F1 scores for each method across multiple domains, illustrating how each approach fares in terms of accuracy and reliability.

Domain	Prec.	Rec.	F1
Infotainment System Control	0.978	0.964	0.971
Vehicle Control	0.994	0.988	0.991
Agent	0.885	0.999	0.939
Navigation and Map Search	0.962	0.935	0.948
Change System Settings	0.954	0.965	0.960
Check Cluster Information	0.994	1.000	0.997
Change Bluetooth Connection	0.945	0.731	0.824
Check Built-in Cam System	1.000	0.909	0.952
Web Portal Search	0.986	0.980	0.983
Vehicle QA	1.000	0.954	0.976
IOT Control	1.000	1.000	1.000
Check the Weather	0.964	1.000	0.982

Table 7: Performance metrics for the Baseline method.

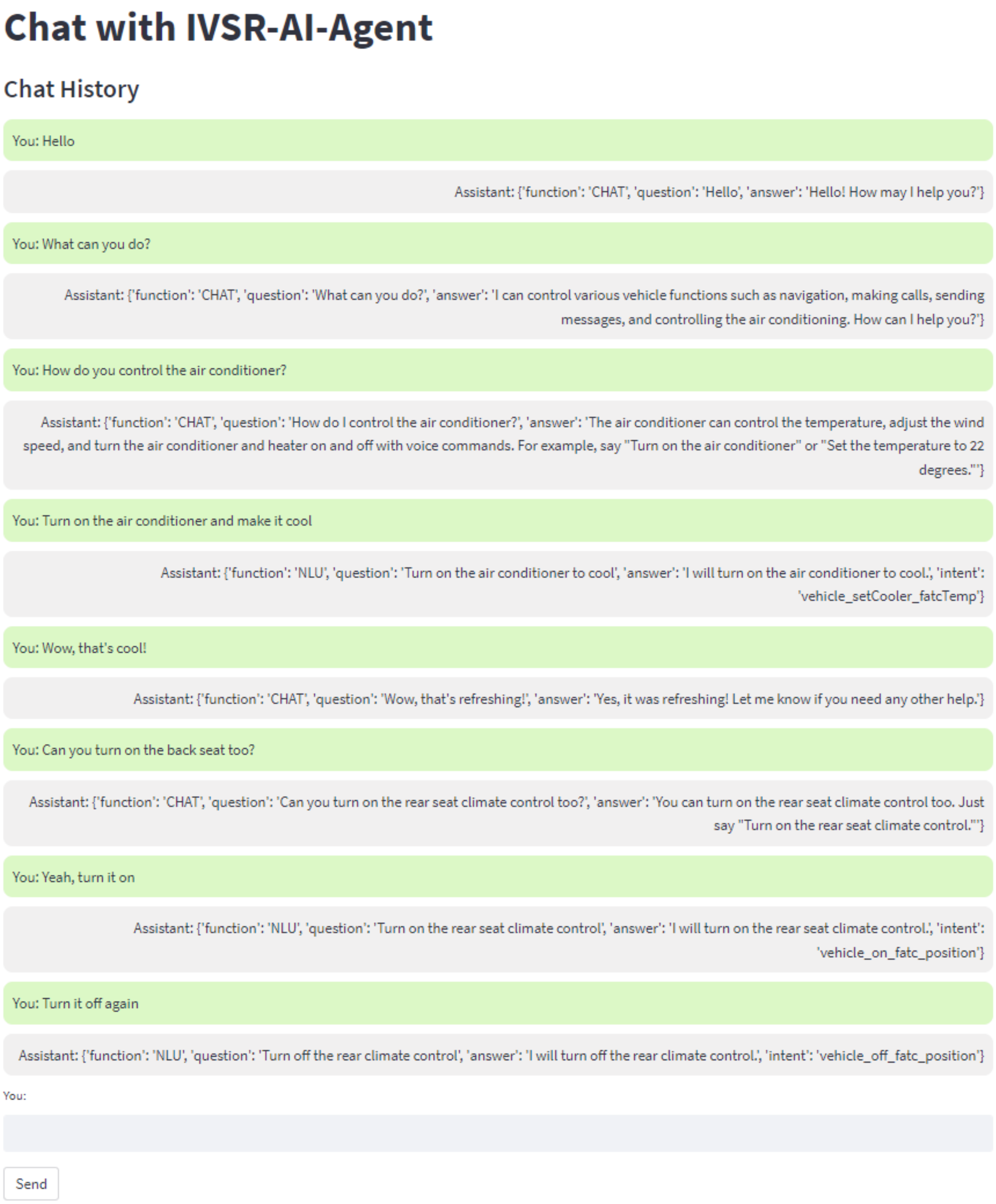
Domain	Prec.	Rec.	F1
Infotainment System Control	0.960	0.978	0.969
Vehicle Control	0.996	0.994	0.995
Agent	0.932	1.000	0.965
Navigation and Map Search	0.974	0.963	0.968
Change System Settings	0.926	0.934	0.930
Check Cluster Information	0.991	0.991	0.991
Change Bluetooth Connection	0.960	0.726	0.827
Check Built-in Cam System	1.000	0.772	0.881
Web Portal Search	1.000	0.993	0.997
Vehicle QA	0.945	0.963	0.954
IOT Control	0.989	1.000	0.994
Check the Weather	0.982	1.000	0.991

Table 8: Performance metrics for the Proposed (GPT-4o) method.

Domain	Prec.	Rec.	F1
Infotainment System Control	0.949	0.842	0.892
Vehicle Control	0.989	0.954	0.972
Agent	0.554	0.934	0.695
Navigation and Map Search	0.894	0.674	0.769
Change System Settings	0.945	0.386	0.548
Check Cluster Information	0.978	0.922	0.949
Change Bluetooth Connection	0.869	0.991	0.926
Check Built-in Cam System	0.938	0.761	0.840
Web Portal Search	0.773	0.115	0.200
Vehicle QA	0.695	0.972	0.811
IOT Control	0.955	0.944	0.949
Check the Weather	0.929	0.481	0.634

Table 9: Performance metrics for the LLM-only (GPT-4o) method.

F Demo Web



Chat with IVSR-AI-Agent

Chat History

You: Hello

Assistant: {'function': 'CHAT', 'question': 'Hello', 'answer': 'Hello! How may I help you?'}

You: What can you do?

Assistant: {'function': 'CHAT', 'question': 'What can you do?', 'answer': 'I can control various vehicle functions such as navigation, making calls, sending messages, and controlling the air conditioning. How can I help you?'}

You: How do you control the air conditioner?

Assistant: {'function': 'CHAT', 'question': 'How do I control the air conditioner?', 'answer': 'The air conditioner can control the temperature, adjust the wind speed, and turn the air conditioner and heater on and off with voice commands. For example, say "Turn on the air conditioner" or "Set the temperature to 22 degrees."'}

You: Turn on the air conditioner and make it cool

Assistant: {'function': 'NLU', 'question': 'Turn on the air conditioner to cool', 'answer': 'I will turn on the air conditioner to cool.', 'intent': 'vehicle_setCooler_fatcTemp'}

You: Wow, that's cool!

Assistant: {'function': 'CHAT', 'question': 'Wow, that's refreshing!', 'answer': 'Yes, it was refreshing! Let me know if you need any other help.'}

You: Can you turn on the back seat too?

Assistant: {'function': 'CHAT', 'question': 'Can you turn on the rear seat climate control too?', 'answer': 'You can turn on the rear seat climate control too. Just say "Turn on the rear seat climate control."'}

You: Yeah, turn it on

Assistant: {'function': 'NLU', 'question': 'Turn on the rear seat climate control', 'answer': 'I will turn on the rear seat climate control.', 'intent': 'vehicle_on_fatc_position'}

You: Turn it off again

Assistant: {'function': 'NLU', 'question': 'Turn off the rear climate control', 'answer': 'I will turn off the rear climate control.', 'intent': 'vehicle_off_fatc_position'}

Figure 6: Interactive Web Demo of Multi-turn Dialogue for IVSR Systems