Benchmarking Contextual Understanding for In-Car Conversational Systems

Philipp Habicht[b,a], Lev Sorokin[a,c], Abdullah Saydemir[a,c], Ken Friedl[a,c], Andrea Stocco[c,d]

[a] *BMW Group, Petuelring 130, Munich, 80809, Germany*
[b] *Humboldt University of Berlin, Unter den Linden 6, Berlin, 10099, Germany*
[c] *Technical University of Munich, Boltzmannstraße 3, Garching, 85748, Germany*
[d] *fortiss GmbH, Guerickestraße 21, Munich, 80805, Germany*

Abstract

In-Car Conversational Question Answering (ConvQA) systems significantly enhance user experience by enabling seamless voice interactions. However, assessing their accuracy and reliability remains a challenge. This paper explores the use of Large Language Models (LLMs) alongside advanced prompting techniques and agent-based methods to evaluate the extent to which ConvQA system responses adhere to user utterances. The focus lies on contextual understanding, the ability to provide accurate venue recommendations considering the user constraints and situational context. To evaluate the utterance/response coherence using an LLM, we synthetically generate user utterances accompanied by correct but also modified failure-containing system responses. We use input-output, chain of thought, self-consistency prompting, as well as multi-agent prompting techniques, with 13 reasoning and non-reasoning LLMs, varying in model size and providers, from OpenAI, DeepSeek, Mistral AI, and Meta.

We evaluate our approach on a case study that involves a user asking for restaurant recommendations. The most substantial improvements are observed for small non-reasoning models when applying advanced prompting techniques, in particular, when applying multi-agent prompting. However, non-reasoning models are significantly surpassed by reasoning models, where the best result is achieved with single-agent prompting incorporating self-consistency. Notably, the DeepSeek-R1 model achieves the highest F1-score of 0.99 at a cost of 0.002 USD per request. Overall, the best trade-off between effectiveness and cost/time efficiency is achieved with the non-reasoning model DeepSeek-V3.

Our results demonstrate that LLM-based evaluations offer a scalable and accurate alternative to traditional human-based evaluations for benchmarking contextual understanding in ConvQA systems.

Keywords: Conversational Systems, Large Language Models, Multi-Agent Systems, Question Answering, Contextual Understanding, Benchmarking

1. Introduction

Conversational Question Answering (ConvQA) systems are becoming increasingly important in various domains, especially in the automotive sector, where voice-controlled systems can significantly enhance safety and convenience. These systems allow users to interact with vehicles using natural language, simplifying tasks such as navigation or controlling vehicle functions [1, 2].

One particular requirement of ConvQA systems is contextual understanding, the system's capability to align suggestions (e.g., restaurant recommendations) with real-time user context, such as location, time, or preferences. For example, if a system provides a venue not matching the user's request (i.e., a restaurant that is too far away or closed), user trust may decline. Given the increasing reliance on in-car systems, ensuring that ConvQA systems work reliably is crucial for both user experience and safety [3, 4, 5, 6].

However, modern ConvQA system are in general LLM-based [4, 7, 8], making them prone to hallucinate and produce incorrect responses to the user, an inherent limitation shared by most learning-based systems [9, 10, 11]. Consequently, ConvQA systems need to be thoroughly tested before they are deployed in cars [12]. However, using human-based evaluation of conversational systems is time-consuming, expensive, and not scalable. In addition, automated metrics such as BLEU [13] or BERT [14], although useful for basic evaluations, are not sufficiently accurate for complex conversational tasks involving contextually rich queries [15]. Also, they require task-specific fine-tuning and are not generalizing well on unseen data, making them less practical to develop [16]. A promising alternative to overcome these challenges lies in using LLMs for evaluation, as they demonstrate strong capabilities in understanding context and conversational dependencies without requiring fine-tuning.

Moreover, existing research in which LLMs evaluate general conversational quality has demonstrated strong alignment with human annotations [17, 18]. Previous work by Friedl et al. [3] has applied LLMs to measure the accuracy of ConvQA systems, providing valuable insights into their potential for automated evaluation. Another study by Giebisch et al [12] focused on the factual relevance and consistency using a small set of selected models. Their work focuses on benchmarking of a judge which is used for evaluation of a retrieval augmented generation (RAG) based conversational system.

Our study provides a comprehensive benchmarking framework, focusing in particular on contextual understanding. For the evaluation of judging methods in combination with LLMs, we generate a manually validated dataset of textual inputs mimicking user inputs to ConvQA, along with positive and negative responses including faults in context-related attributes. We evaluate performance, cost, and time efficiency of 13 LLMs of different sizes, as well as types regarding accessibility and reasoning capability. In addition, we include 6 different prompting methods, including chain of thought prompting for step-by-step reasoning [19] and multi-agent prompting [20, 21]. These methods have proven to better guide LLMs in increasing their reasoning capabilities and significantly improve predictive accuracy [22].

Our study makes the following contributions:

- **Dataset**: We created a synthetic dataset of 600 recommendations for LLM-based judge evaluation. The dataset is human-validated and contains correct and incorrect recommendations to evaluate the abilities of the LLM-based judgment to understand misalignments in recommendations. The methodology behind the creation of dataset and a subset of the dataset are provided for replication: https://github.com/judge-bench

- **Evaluation**: We perform an extensive evaluation of 13 LLMs including reasoning as well as non-reasoning models with 6 prompting techniques and report the effectiveness and efficiency results. In addition, we evaluate which incorrect recommendation can be correctly identified by an LLM-based judgment and which cannot. To foster replicability, we provide prompts for every applied prompting technique in the appendix.

The main findings of our study are:

- Reasoning models outperform non-reasoning models in terms of F-1 scores. Regarding prompting techniques, a negligible effect is observed for reasoning models, while for non-reasoning models, the best improvement is identified for GPT-based models when using advanced prompting. Among the evaluated models, DeepSeek-R1 achieved the highest overall performance with an F-1 score of 0.99, while the best-performing non-reasoning model, DeepSeek-V3, reached an F-1 score of 0.98. Smaller reasoning models, such as o4-mini and o3-mini, consistently surpassed F-1 scores of 0.90, indicating that reasoning capabilities can compensate for smaller model sizes.

- Non-reasoning models show a high performance variability depending on the prompting strategy employed, with advanced techniques generally leading to improved outcomes. However, multi-agent prompting yields inconsistent results, particularly for reasoning models, performing similarly or even worse than single-agent-based prompting.

- In terms of efficiency, the Llama-405B model with self-consistency prompting represents the most resource-intensive configuration (approximately 50 seconds per evaluation), while Mistral with basic I/O prompting demonstrates the fastest response time (approximately 1 second). The most cost-efficient reasoning model is DeepSeek-R1, with an average cost of 0.002 USD per evaluation, while the most cost-efficient non-reasoning model is DeepSeek-V3 with 0.001 USD per evaluation offering at the same time the best trade-off between cost, latency, and performance.

The paper is structured as follows. Section 2 presents related work. Section 3 introduces our case study, followed by a description of applied prompting techniques in Section 4. In Section 5 we present the dataset. Section 6 outlines the experimental design. Section 7 presents the results and Section 8 discusses their implications. Section 9 addresses potential threats to validity. Section 10 concludes the paper and outlines directions for future research. Additional details on the results and prompts are provided in the appendix.

2. Related Work

Generative systems are commonly evaluated using metrics based on word overlap or similarity, such as BLEU [13] or ROUGE [23]. These metrics typically require gold-standard reference data, which may not always be available. Additionally, they were originally designed for specific tasks: BLEU for machine translation and ROUGE for summarization. These metrics impose rigid expectations on the generated text, offering limited tolerance for variation in phrasing or lexical choice. Although they remain widely used, studies have shown that their correlation with human judgment is weak or negligible [14, 24, 25].

Several papers have outlined LLM-driven evaluation methods [26, 27, 28] and their advantages [29], in particular for text summarization tasks. The idea of applying an LLM for evaluating other LLMs, called *LLM-as-a-Judge*, has been presented initially by Zheng et al. [30]. They have evaluated the judging capabilities of LLMs by using human curated question-answer pairs, including generic multi-turn conversations for categories such as math or text summarization with open-ended questions. Their evaluation shows that LLMs reach an agreement of more than 80% with human experts. However, their study is limited to a small selection of non-reasoning models, does not consider contextual understanding in the automotive context, and does not evaluate advanced prompting techniques.

In the automotive domain, Friedl et al. [3] evaluated LLM-based judges for in-car-based conversational information retrieval. Multiple personas were created to let the LLM judge whether the system response fits to user question. Both questions and answers were crafted by human experts. Their evaluation involving three LLM models with zero-shot prompting as well as multi-persona and max-vote prompting, shows that LLM-based judgment achieve up to 94% agreement with human experts. However, the evaluation was only performed on follow-up question answering, implicit understanding, and handling harmful user inputs.

Giebisch et al. [12] evaluated LLM-based-judgment regarding the factual correctness in the in-car context. They have shown that LLM-based judgment achieves up to 90% agreement with human experts. However, their study is limited to evaluating requests for the properties of factual correctness and consistency for a RAG-based system [24]. Our work, on the other hand, focuses on the contextual understanding and provides an extensive study including small but also large scale conventional as well as reasoning models. In addition, it is independent of the underlying conversational system technique.

Conversational datasets such as CoQA [31], MMDialog [32], and VACW [33], and datasets such as MultiWOZ2.2 [34], KVERT [35], which in particular focus on navigational requests and recommendations, do exist. However, these datasets do not incorporate all contextual parameters such as time, cost, and

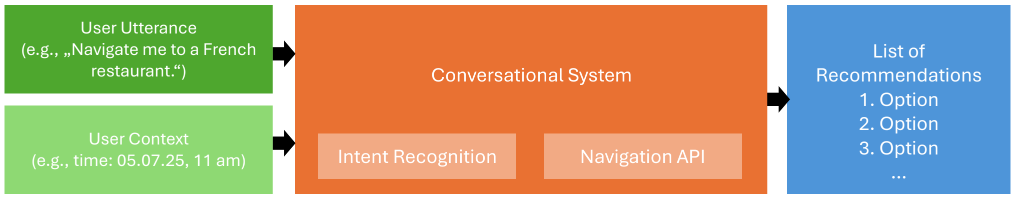


Figure 1: Example of a ConvQA for navigation purposes. The left hand side depicts the system inputs, while the right hand side depicts its outputs with venue recommendations.

location in a single request, nor simulate incorrect system responses which are required for our contextual understanding benchmark.

3. Motivation

In the following, we motivate our work by introducing our case study of benchmarking conversational question answering systems (ConvQA).

ConvQA systems are a branch of conversational AI designed to understand and respond to user input in multi-turn dialogues. These systems incorporate functionalities such as information retrieval, API calls, and web searches to handle complex user interactions [1, 2]. ConvQA systems are applicable across various domains, with one significant area being the automotive industry. In vehicles, these systems facilitate natural language interactions with core functions such as navigation, access to the car manual, and control of car features like audio settings or climate control [36, 12].

Figure 1 illustrates a ConvQA system integrated into a vehicle for navigation purposes.

An example workflow in processing the user request is as follows: first, the system captures the user input, which is, in general, provided in the form of a speech utterance. In the next step, the speech utterance is converted into text, and in following analyzed by the intent recognizer. The intent recognizer tries to identify an appropriate tool to process the request further. In our example, the request is handled over to the navigational tool, which collects first contextual data such as the users location and the time of the request, followed by calling the navigational API, which provides location information

User Block

- **Location**:
  - Latitude: 52.51551195963503,
  - Longitude: 13.457670999834445,
  - Description: Berlin, Friedrichshain
- **Date**: Mon, 06 Mar 2024
- **Time**: 20:35
- **User Utterance**: Hey, can you find me a high-end luxury French food restaurant that's highly rated from 4.5 upwards

System Block

- **Name**: Le Petit Paris
- **Location**:
  - Latitude: 52.51751195963503,
  - Longitude: 13.457670999834445,
  - Description: Berlin, Friedrichshain
- **Cost**: high
- **Opening Hours**:
  - Monday: 10:00 – 20:00
  - Tuesday: 12:00 – 22:00
  - Wednesday: 12:00 – 22:00
  - Thursday: 12:00 – 22:00
  - Friday: 12:00 – 22:00
  - Saturday: 12:00 – 22:00
  - Sunday: Closed
- **Cuisine Type**: French
- **Menu**: Foie Gras, Coq au Vin, Ratatouille, Crème Brûlée, Wine
- **Rating**: 4.6
- **Distance in km**: 1.334
- **Duration in min**: 8

Figure 2: Information passed to the judge under test: the user block with the request for a high-end French restaurant at 20:35, and an example of a system response, where the system recommends a non-suitable venue, where opening hours end on Monday at 20:00.

of relevant venues. The system then selects some nearby venues, considering the user preferences provided in the request.

For the remainder of this paper, we denote the user utterance combined with its contextual data such as location, data and time as *user block*. We denote the response that the system generates as *system block*. The response contains parametrized and detailed output of venue information, including attributes such as *name*, *location*, *cost* or *opening hours*, *rating*, and *cuisine*.

Examples of a user block and system block are shown in Figure 2. In this example, all venue information in the system block align with the users request, except for the opening times. I.e., the proposed venue is closed at the time when the user sends the request.

The goal of this study is to evaluate LLM-based judgment techniques that assess the quality of a system response. Specifically, how well the system block aligns with a given user block.

4. Prompting Techniques

In our study, we evaluate different prompting techniques when applying LLMs for judgment. Prompt engineering has become a crucial method for enhancing the performance of LLM. By providing task-specific instructions,

Table 1: Overview of applied prompting techniques, with brief descriptions and variation used (e.g., number of agents or outputs).

Name	Description	Variation
Input-Output (I/O)	Passing one prompt to one model without examples	1
Chain of Thought (CoT)	Give n examples for reasoning about the task alongside the prompt	1, 3, 5
Self-Consistency (SC)	Multiple CoT reasoning paths; selection of most frequent verdict	3, 5
Multi-Agent Base (MAB)	Agents (same LLM) with distinct roles discuss verdict for one round	1
Multi-Agent Debate (MAD)	Agents (same LLM) with distinct roles discuss verdict until agreement, for a given max. number of rounds	3
Agent Roundtable (AR)	Agents of different LLMs deliberate until agreement, for a given max. number of rounds	3

prompting allows models to adapt to various downstream tasks without modifying their core parameters. Instead of retraining or fine-tuning the model, prompts are designed to give context and steer LLMs toward desired behaviors [37, 38]. As shown in previous studies [3, 12], the prompting method can have a significant impact on the performance of a model. An overview of all prompting techniques used in this study is provided in Table 1 and explained in the following in detail.

4.1. Input-Output Prompting

Input Output (I/O) prompting, as outlined in [39], is the most standard technique for prompting LLMs. Given an input or prompt x, the goal is to generate a corresponding output y. This method leverages the autoregressive properties of transformer models, which sequentially map x to y by estimating the conditional probability distribution:

$$\mathbb{P}(y \mid x)$$

To generate the actual output y^*, the model selects the most probable output by maximizing:

$$y^* = \arg\max_y \mathbb{P}(y \mid x)$$

For a practical example of I/O prompting and the prompt templates used, please refer to Figure A.10 in the Appendix.

4.2. Chain of Thought

Chain of Thought (CoT) prompting is a technique developed to enhance the reasoning capabilities of language models by guiding them to generate structured, step-by-step thought processes [19]. Unlike I/O prompting, which directly maps an input x to an output y, CoT prompting encourages the LLM to produce intermediate reasoning steps $z = (z_1, z_2, \dots, z_M)$ that lead to the final answer y. Each step z_i, where $i = 1, \dots, M$, represents a logical inference or thought that connects x to y, effectively mimicking human problem-solving methods by breaking down complex tasks into manageable sub-steps.

To implement CoT prompting, we employ few-shot (n-shot) prompting by providing the model with N examples, also known as shots. Each example e_j, for $j = 1, \dots, N$, consists of an input x_j, a corresponding sequence of reasoning steps $z_j = (z_{j_1}, z_{j_2}, \dots, z_{j_M})$, and an output y_j. These examples guide the model in generating reasoning steps for new inputs.

Given an input x, the model aims to generate appropriate reasoning steps z and produce the final output y, leveraging the provided examples $E = \{e_1, e_2, \dots, e_N\}$. The probabilistic formulation of this process is expressed as:

$$\mathbb{P}(y, z \mid x, E) = \mathbb{P}(z \mid x, E) * \mathbb{P}(y \mid x, z, E)$$

To generate the final output, the LLM selects the output that maximizes the joint probability:

$$(y^*, z^*) = \arg\max_{y,z} \mathbb{P}(y, z \mid x, E)$$

CoT prompting has been shown to significantly improve the accuracy of language models in tasks that require common sense, mathematical, and symbolic reasoning compared to the more straightforward I/O prompting. By incorporating a few-shot examples, the model learns to generalize the reasoning process to new inputs, enhancing its problem-solving capabilities. For a practical example of CoT prompting and the prompt templates used, please refer to Figure A.11 in the Appendix.

4.3. Self-Consistency

Self-Consistency (SC) builds upon CoT prompting by addressing the variability and randomness inherent in the reasoning process of LLMs. Due to the non-deterministic nature and the diversity of possible reasoning paths, the model may produce different outputs y, when prompted identically. To mitigate this, SC involves independently sampling multiple reasoning paths $\{z_1, z_2, \ldots, z_n\}$ for a given input x, effectively generating a diverse set of candidate answers $\{y_1, y_2, \ldots, y_n\}$. The underlying assumption is that correct reasoning processes are more likely to converge on the same answer, whereas incorrect or flawed reasoning will result in a wider variety of answers. In practice, for each sampled reasoning path z_i, the model produces a corresponding output y_i. The process can be formulated as:

$$(y_i, z_i) = \arg\max_y \mathbb{P}(y, z \mid x, E)$$

where E represents the set of few-shot examples used in CoT prompting. Here, y_i and z_i are jointly sampled from the probability distribution conditioned on the input x and the examples E. After obtaining multiple outputs, the final answer is determined by aggregating these outputs and selecting the most frequent among them:

$$y^* = \text{mode}\,(y_1, y_2, \ldots, y_n)$$

This method improves the accuracy of language models in tasks involving arithmetic, commonsense reasoning, and other complex problem-solving scenarios, as it mitigates the risks associated with a single, potentially flawed reasoning path [40].

4.4. Multi-Agent Systems

Building on these fundamentals, agentic methods were developed where agents with specific characteristics are able to collaborate to make decisions. Inspired by the *society of minds* concept [41], this approach highlights agent communication for tackling complex tasks [42]. In the following, we will explain how we have applied these methods in our research, whereby there may be deviations from the original implementations.

4.4.1. Multi-Agent Base

To further enhance decision making capabilities, we adopted a multi-agent approach as outlined in [3]. In this method, here called Multi-Agent Base (MAB), a set of agents tailored to the key performance indicators (KPIs) was defined. These agents independently make decisions based on the input, and the final outcome is determined by aggregating their responses.

Let x be the given input, and let $\{A_1, A_2, \ldots, A_M\}$ denote the set of M agents. Each agent A_i is characterized by its own persona in a prompt P_i, which influences its interpretation and processing of the input x. Furthermore, each agent A_i generates reasoning steps z_i and a corresponding output y_i, resulting in:

$$\mathbb{P}_{A_i}(y, z \mid x) = \mathbb{P}(y, z \mid x, P_i)$$

Each agent A_i generates its reasoning and outputs by maximizing:

$$(y_i, z_i) = \arg\max_{y,z} \mathbb{P}_{A_i}(y, z \mid x)$$

The set of responses from all agents is $\{(y_1, z_1), (y_2, z_2) \ldots, (y_M, z_M)\}$. To arrive at the final decision y^*, we aggregate the outputs y_i using mode:

$$y^* = \text{mode}\,(y_1, y_2, \ldots, y_M)$$

The rationale is that leveraging agents with diverse perspectives leads to sampling from different parts of the probability distribution, resulting in varying responses. Further, when agents independently reach the same conclusion, it boosts confidence in the decision. Aggregating their responses creates a more balanced and robust outcome by reducing individual biases or errors. Prompt templates and examples for the agents will be shown and further discussed in the next chapter.

4.4.2. Multi-Agent Debate

The agentic approach by [21], called Multi-Agent Debate (MAD) in this study, extends the MAB method by allowing multiple rounds of discussion among agents. Instead of each agent independently producing a decision, agents are able to collaborate over several rounds, refining their responses based on the insights of others, if no agreement was found earlier. This iterative process leads to improved accuracy, reduced bias, and better handling of uncertainties, as agents collectively address ambiguities and refine their decisions [20]. MAD consists of three phases:

Phase 1: Initial Response Generation. Let x be the input, and let $\{A_1, A_2, \ldots, A_M\}$ denote the set of M agents, each defined by a specific persona in a prompt P_i. Based on its unique characteristics, each agent A_i generates initial reasoning steps $z_i^{(0)}$ and an initial response $y_i^{(0)}$:

$$(y_i^{(0)}, z_i^{(0)}) = \arg\max_y \mathbb{P}_{A_i}(y \mid x)$$

Phase 2: Multi-Round Discussion. If no agreement is reached after the initial round, agents exchange their reasoning and outputs and engage in discussions. In each subsequent round $r = 1, 2, \ldots, R$, each agent A_i refines its response by considering the input x, its own previous reasoning and response $(y_i^{(r-1)}, z_i^{(r-1})$, and the reasoning and responses from all other agents in the previous round $\{(y_j^{(r-1)}, z_j^{(r-1)}) \mid j = 1, 2, \ldots, M; j \neq i\}$, resulting in:

$$(y_i^{(r)}, z_i^{(r)}) = \arg\max_{y,z} \mathbb{P}_{A_i}(y, z \mid x, y_i^{(r-1)}, z_i^{(r-1)}, y_j^{(r-1)}, z_j^{(r-1)})$$

This process continues for up to R rounds or until a consensus is reached. Consensus is achieved when all agents A_i agree on the same output:

$$y_1^{(r)} = y_2^{(r)} = \cdots = y_M^{(r)} = y^*$$

Phase 3: Final Decision. If consensus is not reached after R rounds, the final decision y^* is determined by majority voting among the responses in the last round:

$$y^* = \text{mode}(y_1^{(R)}, y_2^{(R)}, \ldots, y_M^{(R)})$$

MAD approach operates on the hypothesis that through iterative discussions, agents can converge toward a more accurate decision by learning from reasoning of others. Engaging in multiple rounds allows agents to address ambiguities, correct misunderstandings, and collectively refine their responses.

4.4.3. Agent Roundtable

Agent Roundtable (AR) approach by [43] differs from MAD method by two key components. First, instead of employing multiple agents within a single model, it utilizes CoT prompting across different LLMs L_i, such as GPT-4o, Mistral-Nemo, and DeepSeek-R1. This mitigates the risks of inherent model biases, limited knowledge scopes, and the absence of external feedback that could arise if all answers were generated by the same data and model architectures. Second, it utilizes uncertainty confidence estimation prompts, as introduced by [44], to allow each agent to assess and express its confidence in the correctness of its response, enabling more confident responses to guide the discussion and making it easier to resolve uncertainties and reach consensus. AR method also consists of three phases:

Phase 1: Initial Response Generation. Given the input x and a set of few-shot examples E for CoT prompting, each LLM L_i generates initial reasoning steps $z_i^{(0)}$, an initial response $y_i^{(0)}$ and an associated confidence level $p_i^{(0)} \in [0,1]$ of how confident the LLM is about its correctness of its decision y:

$$(y_i^{(0)}, z_i^{(0)}, p_i^{(0)}) = \arg\max_{y,z,p} \mathbb{P}_{L_i}(y, z, p \mid x, E)$$

Phase 2: Multi-round Discussion. If no agreement is reached after the initial round, the LLMs exchange their reasoning, outputs, and confidence levels. In each subsequent round $r = 1, 2, \ldots, R$, each LLM L_i refines its reasoning, output and confidence score:

$$(y_i^{(r)}, z_i^{(r)}, p_i^{(r)}) = \arg\max_{y,z,p} \mathbb{P}_{L_i}(y, z, p \mid x, E, y_i^{(r-1)}, z_i^{(r-1)}, p_i^{(r-1)}, y_j^{(r-1)}, z_j^{(r-1)}, p_j^{(r-1)})$$

This process continues for up to R rounds or until a consensus is reached, defined as the point when all LLMs L_i agree on the same output:

$$y_1^{(r)} = y_2^{(r)} = \cdots = y_M^{(r)} = y^*$$

Phase 3: Final Decision. At the end of the final round R, the final decision y^* is determined using a confidence-weighted voting scheme. First, each LLMs confidence $p_i^{(R)}$ is calibrated using a function $f(p_i^{(r)})$. Calibrated confidence scores are used as weights to compute the final answer:

$$y^* = \arg\max_y \sum_{i=1}^{M} f(p_i^{(r)}) \cdot 1(y_i^{(R)} = y)$$

where y is a distinct output generated by any of the LLMs (e.g. true or false), $p_i^{(r)}$ is the original confidence of the LLMs L_i, and $f(p_i^{(r)})$ is the calibrated confidence defined as:

$$f(p_i^{(r)}) = \begin{cases} 1.0 & \text{if } p_i^{(r)} = 1.0 \\ 0.8 & \text{if } 0.9 \leq p_i^{(r)} < 1.0 \\ 0.5 & \text{if } 0.8 \leq p_i^{(r)} < 0.9 \\ 0.3 & \text{if } 0.6 \leq p_i^{(r)} < 0.8 \\ 0.1 & \text{otherwise} \end{cases}$$

Transformation of LLM confidence levels is necessary because they often struggle with accurately interpreting numerical relationships in rankings; therefore, calibration helps aligning their confidence scores with their actual performance.

5. Dataset Generation

To evaluate LLMs for contextual understanding, we have developed a dataset in collaboration with our industrial partner BMW. The dataset consists of 100 synthetically generated user requests (i.e., user blocks) and 600 system blocks simulating a conversational system's response. Each user block combines natural language with contextual metadata such as time, date, and location, as well as semantic preferences regarding cuisine, price, and rating, simulating a user's input.

For every user block, the dataset includes one correct system block (recommendation) where all parameters are aligned and five incorrect recommendations with misalignments based on induced errors. By having misaligned recommendations, the goal is to validate whether the LLMs under evaluation can reliably detect discrepancies in the responses, which is relevant when applying the judging technique later for the detection of failures in conversational systems. In the following, we explain in detail how the user and system blocks are generated automatically.

5.1. User Block Generation

To generate diverse in-car user-system interactions for contextual understanding, we follow a structured procedure as outlined in Algorithm 1. The generation begins by initializing an empty list of user blocks (line 1) and iterating 100 times to create a new user block. To generate a new user block, the algorithm samples uniformly a location from a predefined list of ten urban areas across Berlin and Munich (line 3).

Each location is represented by a coordinate pair and a district label (e.g., "Prenzlauer Berg, Berlin") to allow for downstream geographic relevance evaluation. Next, a calendar date d is randomly drawn from a list of dates $\mathcal{D}$ including dates in the year 2024 (line 4). The algorithm then samples a time t uniformly from the range 08:00 to 22:00 (line 5), covering dining hours during which users request restaurant recommendations while driving.

After establishing the contextual frame, the algorithm proceeds to define the user's preferences. It samples a cuisine type c from a set of 20

Algorithm 1: Generation of user blocks with contextual information

Input : $\mathcal{L}$: List of 10 urban locations (in Berlin, Munich)
$\mathcal{D}$: Set of calendar dates from 2024
$\mathcal{T}$: Time range between 08:00 and 22:00
$\mathcal{C}$: Set of 20 cuisine types with 5 lexical variants each
$\mathcal{K}_{\text{cost}}$: Cost categories (low, medium, high) with 15 paraphrases each
$\mathcal{R}_{\text{phrases}}$: Expressions for ratings above 3.5 (e.g., "above 3.8", "at least 4.5")

Output: $\mathcal{U}_{\text{blocks}}$: Set of 100 user blocks

1 $\mathcal{U}_{\text{blocks}} \leftarrow \emptyset$
2 **for** $i \leftarrow 1$ **to** 100 **do**
3 $\quad l \leftarrow$ `sampleUniform`$(\mathcal{L})$
4 $\quad d \leftarrow$ `sampleUniform`$(\mathcal{D})$
5 $\quad t \leftarrow$ `sampleUniform`$(\mathcal{T})$
6 $\quad (c, c_{\text{lex}}) \leftarrow$ `sampleCuisineWithKeyword`$(\mathcal{C})$
7 $\quad k_{\text{cost}} \leftarrow$ `sampleParaphrase`$(\mathcal{K}_{\text{cost}})$
8 $\quad r \leftarrow$ `selectRatingPhrase`$(c, k_{\text{cost}}, \mathcal{R}_{\text{phrases}})$
9 $\quad u_{\text{utt}} \leftarrow$ `generateUtterance`$(c_{\text{lex}}, k_{\text{cost}}, r)$
10 $\quad$ `ctx` $\leftarrow$ `formatContext`$(l, d, t, c, k_{\text{cost}}, r)$
11 $\quad u_{\text{block}} \leftarrow$ `merge`$(u_{\text{utt}},$ `ctx`$)$
12 $\quad \mathcal{U}_{\text{blocks}} \leftarrow \mathcal{U}_{\text{blocks}} \cup \{u_{\text{block}}\}$
13 **return** $\mathcal{U}_{blocks}$

international options (e.g., Italian, Korean, Brazilian), and one of five lexical variants c_{lex} associated with the selected cuisine (line 6). For instance, *Sushi* or *Ramen* might be selected as a keyword for the Japanese category, to ensure lexical diversity. Further, a cost level (low, medium, or high) is defined next (line 7), along with a corresponding paraphrased expression that reflects natural user language, such as *rock-bottom prices* or *luxurious prices*.

We pass both the cost level and the paraphrased expression to GPT-4 to generate a rating phrase from a predefined set of expressions (e.g., *above 3.8*) (line 8). This conditional selection targets to maintain logical consistency, avoiding unrealistic combinations such as *dirt-cheap* with *five-star rating.* All ratings are restricted to values above 3.5, aligned with BMW's internal quality classification standards. With the user preferences defined, we pass the information to GPT-4 to generate an utterance u_{utt} that combines the selected cuisine keyword, cost phrase, and rating expression (line 9).

In the remaining steps of the algorithm, we combine both the user utter-

Algorithm 2: Generation of correct recommendations

Input : $\mathcal{U}$: Set of user blocks
Output: $\mathcal{D}_{\text{pos}}$: Set of correct recommendations

1 $\mathcal{D}_{\text{pos}} \leftarrow \emptyset$
2 **foreach** $u \in \mathcal{U}$ **do**
3 $\quad p \leftarrow \texttt{constructPrompt}(u, \texttt{fully\_aligned})$
4 $\quad r \leftarrow \texttt{GPT4.generate}(p)$
5 $\quad \mathcal{D}_{\text{pos}} \leftarrow \mathcal{D}_{\text{pos}} \cup \{(u, r)\}$
6 **return** $\mathcal{D}_{pos}$

ance with contextual information into a user block and store it in a list (line 11-12). An example of such a generated utterance is: *"Hey, can you find me a high-end luxury French restaurant with a rating of at least 4.5?"*. After completing all 100 iterations, the algorithm returns the set of user blocks.

5.2. System Block Generation

For each user block, we generate six system blocks representing restaurant recommendations. One of these recommendations is a *positive case* (i.e., a fully aligned recommendation), while the remaining five recommendations are *error cases*, each containing exactly one misalignment in one of the following dimensions: location, time, cuisine, cost, or rating. All other parameters are kept identical to isolate the effect of the specific discrepancy. The generation process is illustrated in Algorithm 2 for positive cases and in Algorithm 3 for error cases.

Correct Recommendations. In Algorithm 2, the process begins by initializing an empty list of aligned system blocks $\mathcal{D}_{\text{pos}}$ (line 1). For each user block $u \in \mathcal{U}$ (line 2) (i.e., the ones generated in Algorithm 1), a prompt p is constructed using a template where all context parameters do fully align (line 3). GPT-4 is then prompted to generate a recommendation (line 4), which is stored together with its corresponding user block in (line 5). After processing all user blocks, the algorithm returns the complete set of positive blocks (line 6).

Incorrect Recommendations. To generate an incorrect recommendation (Algorithm 3), the system iterates over the same user block generated with Algorithm 1, serving as the reference for controlled modification. In particular, the algorithm iterates over each error type and generates for each error type an error-specific prompt to introduce exactly one targeted error, e.g., by

Algorithm 3: Generation of recommendations with errors

Input : $\mathcal{U}$: Set of user blocks
$\mathcal{D}_{\text{pos}}$: Fully aligned recommendations
$E = \{\texttt{location}, \texttt{time}, \texttt{cuisine}, \texttt{cost}, \texttt{rating}\}$
Output: $\mathcal{D}_{\text{err}}$: Dictionary of error-specific recommendation sets

```
1  foreach u ∈ U do
2      r_base ← lookup(D_pos, u)
3      foreach e ∈ E do
4          p_e ← constructPrompt(u, e)
5          repeat
6              (e ≠ location) or t > 15
7          until  r_e ← GPT4.generate(p_e)
8          if e = location then
9              t ← MapboxAPI.estimateTravelTime(u.location, r_e.location)
10
11         D_err[e] ← D_err[e] ∪ {(u, r_e)}
12 return D_err
```

asking for a different cuisine than given in the request (line 4). The prompt is then passed to GPT-4 to generate a faulty recommendation r_e (line 6).

Location errors are treated in a specific way: we perform a post-processing and make an external call to the Mapbox API [45] to compute the driving time between the original user location and the generated restaurant location (line 8). If the travel time is below the 15-minute threshold, a new recommendation is generated until the constraint is satisfied. All error-specific recommendations are then stored and returned at the end of the algorithm (line 12). For instance, Figure 2 illustrates a user and system block pair with a time error generated with this approach. For each error case, all parameters except for the error-related parameter remain identical to the corresponding positive case, to ensure that we can evaluate later the judgments performance sensitivity to specific error categories.

Finally, after automatic generation, we passed all system blocks to three domain experts for review to validate the correctness of the prompts. From in total of 600 generated requests, approximately 30% samples required manual refinement due to inaccuracies in GPT-4 generations because of incorrectly aligned or incorrectly misaligned parameters.

6. Experimental Design

In this section, we describe the experimental setup for the evaluation of LLM-based judgement for benchmarking contextual understanding. We describe the research questions, the language models and prompting techniques under test, the evaluation metrics, and the deployment.

6.1. Research Questions

$\mathbf{RQ_0}$ (Data Validation): *What is the validity rate of the benchmarking dataset?*

We evaluate the validity of our benchmarking dataset by performing a user study and passing the system and user blocks to humans who have not been involved in this paper. By evaluating both the validity of in and outputs, and the relation between system and user blocks, we are able to evaluate whether the dataset can be used four LLM-based judge evaluation.

$\mathbf{RQ_1}$ (Effectiveness): *What combination of LLM and method is the most effective for contextual understanding in navigational requests?*

By answering this question, we want to evaluate how well LLMs can identify contextually incorrect as well correct system responses for a given user request.

$\mathbf{RQ_2}$ (Efficiency): *What are the trade-offs between processing time and costs when using different LLMs and methods to benchmark contextual understanding?*

In the second research question, we want evaluate the runtime and costs which are required to perform an evaluation of contextual understanding using different model and method configurations. Runtime and costs are specifically relevant when a large number of tests needs to be executed when testing a ConvQA.

$\mathbf{RQ_3}$ (Failure): *On which misaligned recommendations do LLMs exhibit failures in contextual understanding?*

By answering the third research question, we want to understand which errors in recommendations lead to incorrect judgment results when applying LLMs for evaluation.

6.2. Language Models

For the evaluation of different LLM-based judges using the methods introduced in the previous chapter, we selected diverse LLMs varying in size, accessibility, type, date, and costs. Table 2 provides a detailed overview

Table 2: LLMs used in this study. Note that pricing rates are from the time of the runs and might be different from current rates. Reasoning models are denoted with (R). Token pricing is as of July 2025 and may differ from current rates.

Model Name	**Company**	**Context Window**	**Knowledge Cut-Off**	**Number of Parameters**	**Cost / 1M Input Tokens**	**Cost / 1M Output Tokens**
GPT-3.5 Turbo	OpenAI	16K	Sep 2021	unknown	$0.50	$1.50
GPT-4 Turbo		128K	Dec 2023		$10.00	$30.00
GPT-4o		128K	Oct 2023		$5.00	$15.00
GPT-4.1		1M	Jun 2024		$2.00	$8.00
o3-mini (R)		200K	Oct 2023		$1.10	$4.40
o3 (R)		200K	Jun 2024		$10.00	$40.00
o4-mini (R)		200K	Jun 2024		$1.10	$4.40
DeepSeek-V3	DeepSeek	64K	July 2024	685B	$0.27	$1.10
DeepSeek-R1 (R)		64K	July 2024	671B	$0.55	$2.19
Mistral-Nemo	Mistral	128K	unknown	12B	$0.30	$0.30
Mistral-Large		128K		123B	$3.00	$9.00
Llama-3.1	Meta	128K	Dec 2023	8B	$0.30	$0.61
		128K	Dec 2023	405B	$5.33	$16.00

of the models used in this study, highlighting their key specifications and associated costs. Note that OpenAI charges a discounted rate for batched inputs and DeepSeek charges less for queries during the night. We report the baseline numbers and did not resort to either of these.

6.3. Prompting

For each prompting approach, we define custom prompt templates.

- For I/O we use a prompt template as illustrated in Table 3 without providing examples.
- For SC, we include reasoning paths into the prompt to exemplify reasoning.
- For MAB, we employ three distinct agents with the roles of Investigator, Forensic Examiner, and Auditor. Following the approach of Giebisch et al. [12], we prompted GPT-4 to generate personae with descriptions that may support contextual understanding during evaluation, given their respective qualifications. A complete description of the personae is provided in the Appendix.

Table 3: Prompt template for Input-Output Prompting for the evaluation of whether a system recommendation (system block) fits a user request (user block).

```
You are a critical evaluator tasked with determining whether the information provided by a
car navigation system (System Block) aligns correctly with the user's expressed needs in user
utterance and user context (User Block).

User Block:  <user-block>

Recommendation:  <system-block>

Rules:  <constraints>

Decision:  If any of the above parameters are INCORRECT, the final decision is 'false'.  If
all parameters are CORRECT, the final decision is 'true'.

Please respond strictly following the format specified below:
<output-format>

Make sure the output is always a valid JSON format.
```

- For the remaining prompts we followed guidelines from literature [3] and adopted it to our use case.

Detailed prompt templates for each method and agent definitions can be found in the appendix in Section 10.

6.4. Metrics

$\mathbf{RQ_0}$. To answer RQ_0, we validate the generated dataset along two dimensions. First, we pass a subset, i.e., 20, of user and system blocks to eight independent human annotators, which have been not involved in this work, to rate the validity on a 5-point Likert scale *(1 = invalid, 2 = most likely invalid, 3 = inconclusive, 4 = most likely valid, 5 = valid)*. This step is performed to assess whether the samples represent plausible in-car navigation interactions (Figure B.29, Figure B.30). We use a Likert scale to allow annotators to express graded judgments of validity,

In the second step, we measure the inter-annotator agreement on the user-block system-block annotations, which express whether a system block recommendation is correct (s. Algorithm 2) or incorrect because of a specific error type inserted (e.g., time error, location error, cost error, cuisine error, rating error) (s. Algorithm 3).

We evaluate the correlation with Krippendorff's α [46], a widely applied correlation metric [47, 48] that measures the agreement among annotators in giving the same categorical judgment for the same user–system block pair.

This metric is chosen as it (i) supports an arbitrary number of annotators, (ii) can be applied to nominal as well as ordinal data, and (iii) can robustly handle missing or incomplete annotations.

By assessing both the validity of the system and user blocks, and the inter-rater agreement on the recommendation labels, we are able to evaluate whether our dataset can be reliably used for benchmarking LLM-based judgment.

$\mathbf{RQ_1}$. To evaluate RQ_1, we pass the user-block system block pairs taken from our dataset (s. Section 5) first to the LLM under tests to receive a judgment result in terms of a boolean score, where 1 corresponds to the recommendation being correct and 0 to the recommendation being incorrect. The LLM-based judge is instructed in the following way via prompting to output the score:

- A recommendation is considered incorrect if one of the following applies: a) the location is more than a 15-minute drive away; b) the restaurant is closed at the requested time; c) the cost or rating deviates from the user-specified parameters or d) the cuisine type does not match the request. For ratings, in case the user uses terms such as *around*, a rating is considered as incorrect if the output rating is exceeding a range of 0.2 around the requested rating.
- Otherwise, the recommendation is considered correct.

Finally, the LLM-based judgment result is compared with the underlying label of the corresponding user-block and system block pair. If the results coincide, the judgment is correct; otherwise, it is wrong. Based on the agreement and disagreement results with the actual recommendation alignment scores, we calculate the F-1 score, which is a widely applied metric for the assessment of classification approaches [49]. We apply in particular the F-1 score, because it balances precision and recall, and is therefore more informative than accuracy in the presence of class imbalance, as in our case.

$\mathbf{RQ_2}$. For the evaluation of RQ_2, we measure the time between passing a test to our LLM-based judge and the complete output of its response. In addition, we track the tokens used in the input as well as output prompts. Based on cost-per-token information given by the model provider, we calculate the overall cost per request per model.

$\mathbf{RQ_3}$. To evaluate RQ_3, we evaluate the effectiveness of the judgment technique for each error category applied to the dataset initially, such as location-error, time-error, cuisine-error, cost-error, and rating-error, along with the positive cases to evaluate on which error categories judgment is less effective. Further, we instruct the LLM model to generate an explanation for its judgment to better understand why a particular classification was made.

6.5. Deployment and Request Passing

The models were partially accessed using an API provided as well as were manually deployed in a virtual machine in the cloud. The benchmarking code was executed locally. We used for all prompts a temperature of 0.0 to ensure determinism and gain reliable benchmarking results. Each method was tested on the set of all 600 input-output pairs.

7. Results

In this chapter, we present the results obtained across the experiments for each research question separately.

7.1. Data Validation (RQ_0)

The human evaluation of 20 randomly selected user–system block pairs by eight annotators yielded a mean validity score of 4.1 (out of 5) with a standard deviation of 0.49 for user blocks, and 3.8 (SD = 0.57) for system blocks. To assess inter-rater reliability, we computed Krippendorff's α (ordinal), obtaining 0.73 for user blocks and 0.69 for system blocks, both indicating substantial agreement among annotators. For the recommendation labels, Krippendorff's α (nominal) reached 0.86, reflecting high consistency among raters. Overall, despite being synthetically generated, our dataset demonstrates high validity and substantial human agreement regarding the correctness of its labels.

> RQ_0 (Data Validation). Our benchmarking dataset achieved a mean human-rated validity score of 3.95 out of 5, indicating high overall quality. The recommendation labels for user and system blocks exhibited substantial inter-rater agreement, confirming the dataset's reliability for benchmarking purposes.

7.2. Effectiveness (RQ_1)

The performance results across all methods and prompting techniques are shown in Figure 3. The results show that the combination of advanced prompting methods and larger models yields the best performance in contextual understanding evaluation. In particular, the highest F1-score across all models and techniques of 0.990 is achieved by DeepSeek-R1 with CoT-1 as well as with SC prompting. The worst result is achieved with GPT-3.5 Turbo with the MAD prompting technique. For detailed results, including precision and recall values, we refer the reader to Appendix Appendix B.

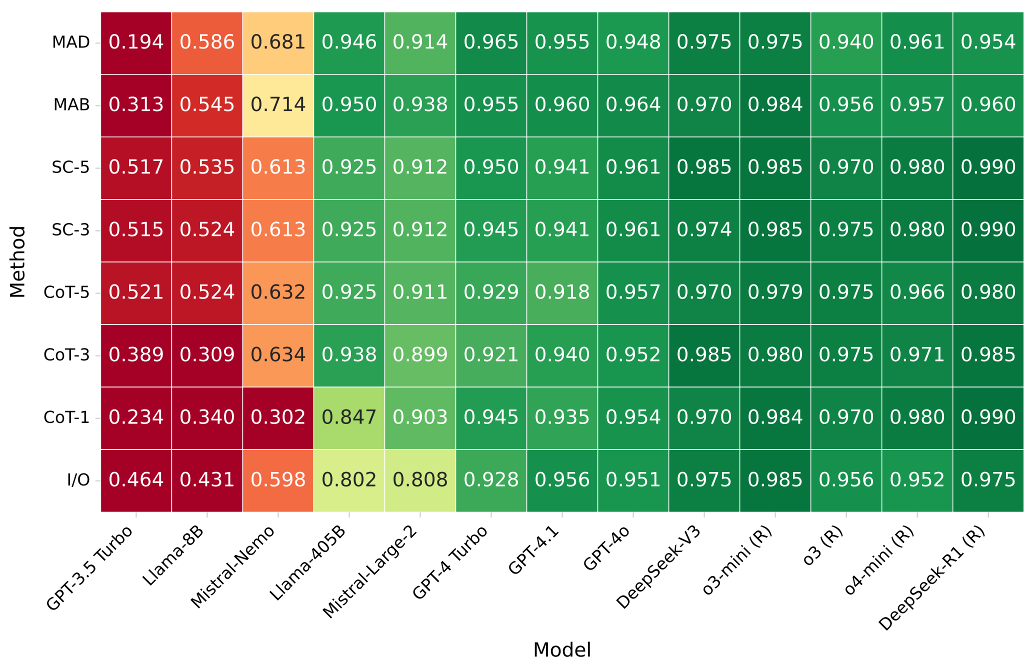


Figure 3: F1-score results for LLMs and prompting techniques evaluated on contextual understanding.

Non-reasoning models. DeepSeek-V3 achieves the best result with an F1-score of 0.985, followed by GPT-4o and GPT-4.1. In general, we observe for non-reasoning models, a high deviation between the performance results across different prompting techniques. Advanced prompting techniques improved results in particular for GPT-based models. However, among the remaining non-reasoning models we could not observe an remarkable effect

Table 4: Precision, recall, and F1-scores for contextual understanding combining multiple models with chain-of-thought prompting.

LLMs	Method	Precision	Recall	F1-score
GPT-3.5 Turbo Mistral-Nemo Llama-8B	AR-CoT-5	0.579	0.840	0.686
GPT-4 Turbo Mistral-Large-2 Llama-405B	AR-CoT-5	0.951	0.980	0.966
o3-mini (R) o4-mini (R) o3 (R)	AR-CoT-5	0.952	1.000	0.976
DeepSeek-R1 (R) DeepSeek-V3 o3-mini (R)	AR-CoT-5	0.970	0.990	0.980

unless that Llama-405B and Mistral-Large-2 show their worst performance with default I/O prompting.

The best result among the smaller non-reasoning models was achieved by Mistral-Nemo using MAB method (0.714). In general, we can observe that small non-reasoning models such as Mistral-Nemo, Llama-8B, and GPT-3.5 Turbo perform significantly worse than larger non-reasoning models such as DeepSeek-V3.

Reasoning models. As for reasoning models we can see that the models almost always perform better than their conventional non-reasoning counterparts. In particular, DeepSeek-R1 and o3-mini show the best performance, where DeepSeek-R1 is achieves slightly higher scores then o3-mini (0.990 vs 0.985). Even small[1] reasoning models such as o4-mini and o3-mini achieve high scores. In general, we can observe that reasoning models achieve F1-scores over 0.90. Large reasoning models consistently delivered high F1-scores, even with simple I/O prompting.

Agent Roundtable with Chain of Thought. Table 4 offers a nuanced view of the performance of different agent combinations for AR-CoT-5 prompting

[1]OpenAI does not expose the exact model sizes but annotates models with mini or nano.

and the balance between precision and recall.

LLM combination that contains only reasoning models, i.e., the second last row from Table 4 performs well and is similar to the group that contains the best reasoning model, best small model, and best large non-reasoning model based on the results in Figure 3. While the latter combinations achieve similar F1-scores, we observe that the group with two DeepSeek models (the last row) produces a more balanced output between precision and recall, whereas the group that consists solely of OpenAI reasoning models yields the maximal recall of 1 with a lower precision score. However, the best AR combination (based on the F1-score), which corresponds to the last group in Table 4 does not perform better then the best single-agent driven model method combination (DeepSeek-R1 with F1-score 0.990, s. Figure 3).

> Effectiveness (RQ_1): DeepSeek-V3 achieves the highest F1-score (0.985) among non-reasoning models, followed by GPT-4o and GPT-4.1, with performance varying significantly across prompting techniques. Among smaller non-reasoning models, Mistral-Nemo performed best (F1 = 0.714), but all smaller models exhibited a clear performance gap compared to larger ones. Reasoning models yielded higher scores than non-reasoning counterparts, with DeepSeek-R1 (F1 = 0.990) and o3-mini (F1 = 0.985) achieving the highest scores.

7.3. Efficiency (RQ_2)

Time efficiency results are visualized in Figure 4. We report here the results of the 10 best-performing models with respect to the F1-score provided in Figure 3. Subfigures show both the number of tokens produced and the average time for a single request.

As we can observe, the best time efficiency is achieved with default I/O prompting with Mistral-Nemo, yielding on average 1 second for one request, while the longest duration was observed with SC-5 with Llama-405B with 50s. These reported prompting methods were, on average across all models, the best/worst regarding time efficiency. We can also observe that the time is in general proportional to the output token count. Furthermore, we can see that reasoning models tend to require more time on average for single requests and produce more output tokens than non-reasoning models, independent of the prompting techniques used.

The cost efficiency results are shown in Figure 5. We calculate the single request cost efficiency based on the cost provided in Table 2. The highest cost for each model was identified with SC-5, while the lowest was with I/O prompting. The lowest-cost of a reasoning model was achieved by DeepSeek-R1 with 0.002 USD (I/O), followed by o4-mini and o3-mini. The most cost-efficient non-reasoning model, as well as the most cost-efficient model overall, was DeepSeek-V3 with a cost of approximately 0.001 USD for one request with I/O prompting.

Efficiency (RQ_2): The fastest setup was Mistral-Nemo with default I/O prompting ($\approx$1 s/request), while the slowest was Llama-405B with SC-5 prompting ($\approx$50 s). Reasoning models were generally slower and produced more tokens. In terms of cost, SC-5 was the most expensive, while I/O prompting was the most economical. DeepSeek-R1 was the most cost-efficient reasoning model ($\approx$USD 0.002/request), and DeepSeek-V3 achieved the lowest overall cost ($\approx$USD 0.001/request), offering the best balance between speed, cost, and performance.

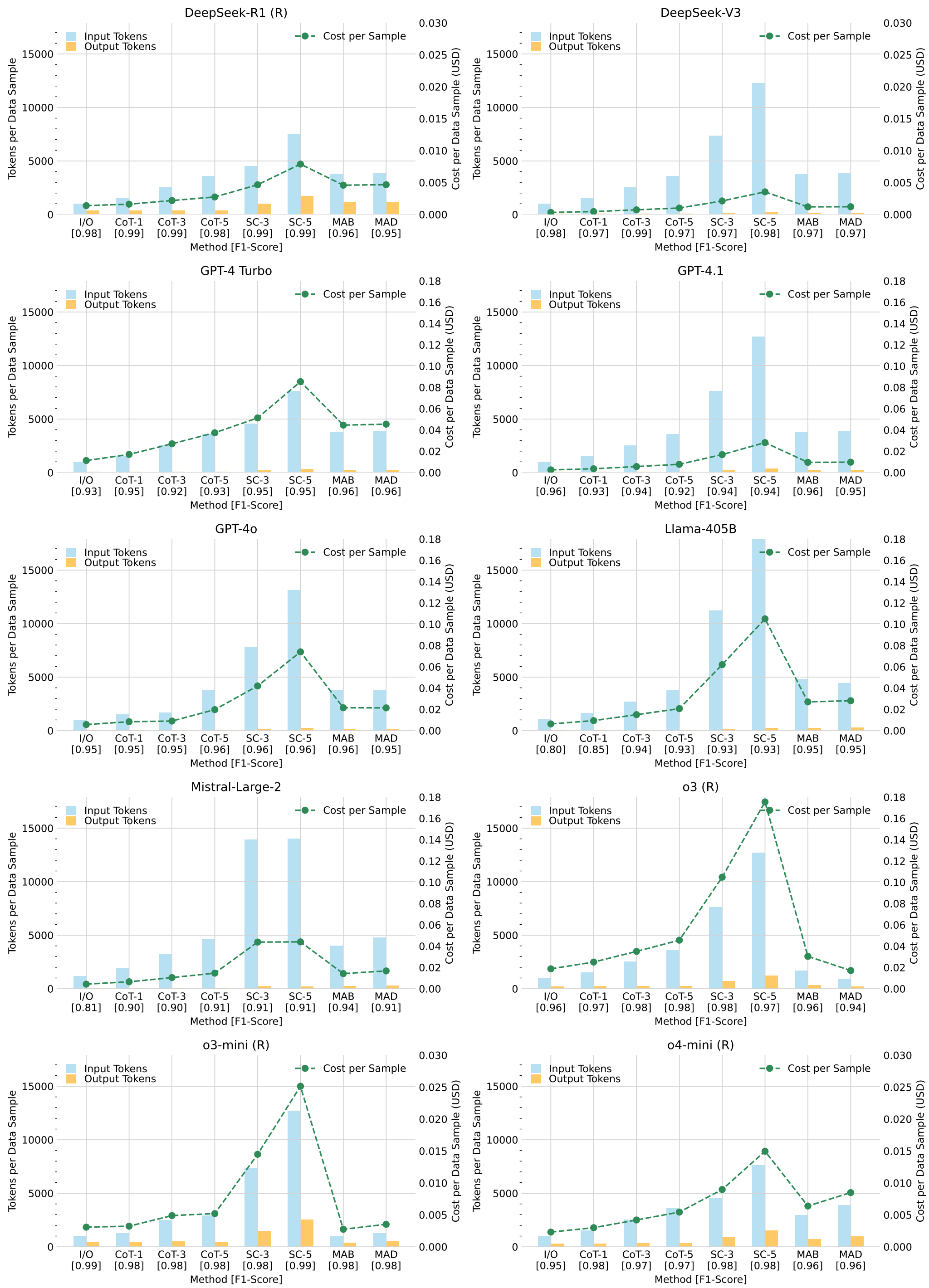


Figure 5: Relationship of evaluated models between Tokens and Cost for Contextual Understanding

When evaluating the tradeoff between effectiveness and cost/time efficiency, we can observe the non-reasoning model DeepSeek-V3 achieves the best balance between performance and time.

7.4. Failure (RQ_3)

In the following, we present an analysis regarding which failure categories (see Section 5.2 for the error category definition) in the data under evaluation were difficult for the models to process and provided an incorrect judgment. An excerpt of the failure categories and model performance per model type and prompting technique is shown in Figure 7 for the worst-performing non-reasoning model, in Figure 8 for the best-performing non-reasoning model, as well as for the best reasoning model in Figure 9. Results of the remaining models are provided in the appendix in **??**.

We can observe that, on average, the most frequent and incorrect judgment with non-reasoning models happened for data which included time or cost errors, as can be seen, for instance, for worst performing non-reasoning model GPT 3.5 Turbo in Figure 7. For reasoning and large non-reasoning models, in general, only data with cost errors yielded accuracy scores below 1 (s. **??**). Overall, we could not observe a relation between prompting technique and error types; however, we observed that advanced prompting, unless round table techniques, produced, in general, slightly better performance for the frequent error categories, time, and cost.

Time errors. For instance, regarding time errors, to evaluate an LLM with AR-CoT-5 involved a request for a *dirt cheap ramen place* at 20:40, but the system suggested a restaurant that was closed at 20:00. Although the restaurant met the preferences of the user in terms of price and distance, the time error remained undetected by the LLM.

Cost errors. For data sets including cost errors, one model failed to detect that the suggested restaurant did not match the cost category given by the user. In one case, a user requested a *top-tier Greek restaurant*, but the system recommended a low-cost option, and the methods AR-CoT-5 did not identify this as an error. By analyzing the prompts, we found out that the LLM judge insists that phrases like *top-tier*, *very luxurious*, *high-end luxurious* do not necessarily mean high-cost; therefore, do not understand these as errors.

In another example, the non-reasoning model GPT-4 Turbo misinterpreted the cost category explaining:

The system provided a restaurant with Portuguese cuisine and a rating of 4.6, which meets the user's requirements. However, the cost category of the restaurant is high, not premium elite as requested by the user. Therefore, the system's information does not align with the user's needs.

This reasoning led GPT-4 Turbo to classify this case as false, convincing Llama-405B and Mistral-Large-2 to judge in the same way in the round table prompting.

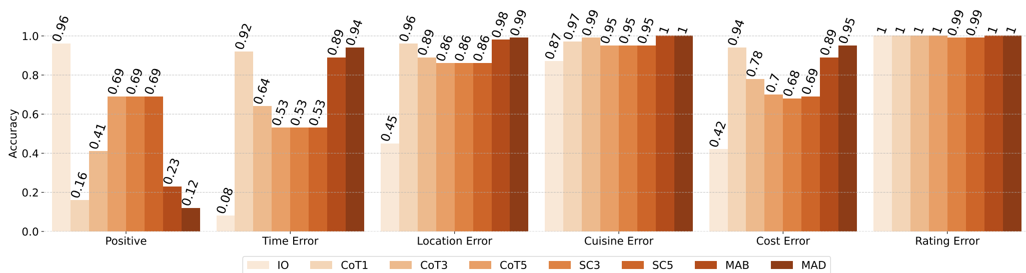


Figure 7: GPT 3.5 Turbo accuracy results based on the category of data used.

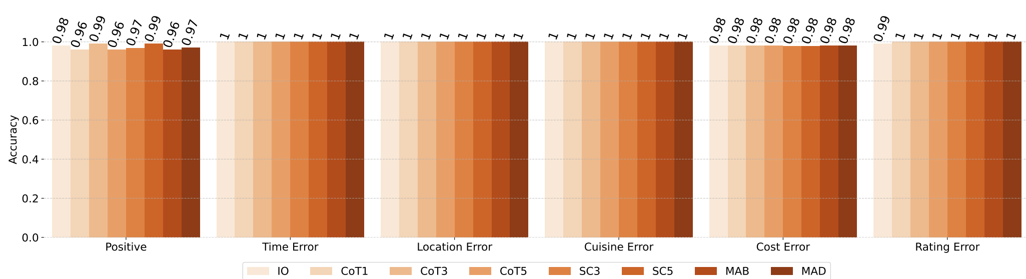


Figure 8: DeepSeek-V3 accuracy results based on the category of data used.

Figure 9: DeepSeek-R1 accuracy results based on the category of data used.

Failure (RQ_3): Non-reasoning models most frequently misjudge inputs containing time or cost errors, with GPT 3.5 Turbo showing the weakest performance. Reasoning models and large non-reasoning models generally achieve a high accuracy across misaligned recommendations, with cost errors being the only category where accuracies drop below 1.

8. Discussion

In the following, we discuss some observations and considerations regarding the application of LLMs for the contextual understanding benchmark.

Importance of Prompting Technique. We observed for GPT models in general, an improvement when using advanced prompting. However, advanced prompting techniques incur higher costs and execution time. However, SC-3 and SC-5 did not show much difference in performance, but led to twice more cost. We recommend that preliminary experiments should be conducted to decide if self-consistency with three repetitions shows already satisfactory results. When time efficiency is relevant, simpler prompting techniques such as CoT are reasonable to be employed in case the default model response time is not already relatively small such as for instance for DeepSeek-V3. Besides the technique, prompt optimization could be applied by leveraging optimization techniques [50].

Usage of reasoning models. Reasoning models significantly improve performance given the results in Section 7 even when combined with simpler prompting techniques. Latest non-reasoning models like DeepSeek-V3 have shown comparable accuracy results to reasoning models. When comparing proprietary reasoning models with open-source reasoning models, we did not see significant performance gaps, unless open-source models incur significantly lower cost.

API usage. We observed that the latency for API calls in the evening is on average, lower. Also, asynchronous calls to APIs for faster processing might be useful to be employed. In longer runs, we also observed interruptions due to connection problems. It is therefore recommended to have a rerun mechanism for longer runs.

Single-agent prompting vs. Multi-agent Prompting. The best F1-score is achieved with DeepSeek-R1 with CoT/SC while for GPT-based non-reasoning models using multiple agents for prompting, achieved the best effectiveness. However, for reasoning models, single-model based prompting such as CoT-3 or SC-5 outperformed all prompting techniques.

Model Size. For non-reasoning models, we observe that performance results improve with increasing model size. However, we cannot conclude this for GPT-based models, as no size information is disclosed from the provider. For reasoning models, we can only rely on the vague size information provided by OpenAI and conclude that the model size does not affect the performance results.

Application to other domains. In our benchmarking, we evaluate conversational interactions for restaurant navigation requests, but the same evaluation approach can be extended to other domains, such as locating fuel stations, bars, or even performing different tasks like starting a radio or opening windows. Several adjustments to the framework would be required.

First, input adaptation: the user request representation must include domain-specific context parameters. For example, in the case of fuel stations, additional attributes such as fuel type or charging availability would be needed, while others, like price range or opening hours, may remain but with adjusted ranges. The prompting schema remains unchanged, except for substituting restaurant-related details with domain-relevant ones.

Second, output adaptation: the system's response format and error categories must be adjusted to the new domain. For instance, attributes like cuisine or rating would be replaced with parameters such as fuel availability, charging speed, or diesel compatibility. The corresponding error taxonomy should also be updated (e.g., fuel type error, availability error, price error), while the evaluation logic matching user intent and system response remains structurally identical.

Finally, the judging mechanism must be adapted to define correctness in the new context. For restaurants, correctness depends on factors like cuisine, distance, rating, and opening times, while for fuel stations it would instead depend on fuel compatibility, availability, operational status, and distance thresholds. The underlying judging mechanism, however, remains unchanged.

9. Threats to Validity

External Validity. Our evaluation is based on a single case study. While this case study originates from a real-world industrial context and is representative for practical usage, generalization of the results to other domains or recommendation tasks is not shown. However, we generate a diverse dataset regarding the linguistic expression of the user intent as well as the information in the request. Future work should investigate additional domains to support broader claims. However, our case study contains contextual data which should be similar for other case studies to benchmark contextual understanding.

Internal Validity. Due to the non-determinism of large language models, results may vary across runs. We controlled for this by setting the temperature to 0 wherever possible; however, for OpenAI reasoning models, temperature control is not supported, and we used the default setting. Additionally, we performed prompt optimization and evaluated the determinism of the generated outputs manually on a small ratio of the dataset in preliminary experiments. Regarding the dataset validity, a human-based validation was performed to mitigate the bias of having generated incorrect user block, system block combinations as explained in Section 5.

Construct Validity. Our study relies on proprietary models from OpenAI, for which in particular the model size is not publicly shared. We cannot therefore make any conclusions regarding the size of the model in connection with its performance. While this affects reproducibility, we use widely accessible APIs and standard configurations to ensure that our setup can be replicated.

10. Conclusion and Future Work

In this paper, we presented a comprehensive study on evaluating LLM-based benchmarking for in-car ConvQA systems, focusing on contextual understanding in navigation tasks. Considering advanced open-source, closed-source LLMs and sophisticated prompting as well as agent-based techniques, the study presented an alternative to human-based evaluation.

Our results show that combining advanced prompting techniques can improve the accuracy of LLM models for contextual benchmarking, leading up to F1-scores close to 1. The biggest improvement can be achieved, in particular for non-reasoning models. Multi-agent prompting techniques do improve the effectiveness for non-reasoning models, while for reasoning models best results were achieved with single-agent prompting with self-consistency. However, the best overall tradeoff between cost/time efficiency and effectiveness is achieved with the non-reasoning model DeepSeek-V3.

To provide a more comprehensive assessment, future work could include other places of interest besides restaurants, such as gas stations, electric vehicle chargers, or grocery stores. It would also be interesting to understand whether our findings apply also for recommendations in other languages which is in particular important when deploying vehicles with conversational assistants in different countries. Moreover, increasing the complexity of human-machine interactions by incorporating multi-turn conversations, rather than single-turn queries, would offer deeper insights into the ability to manage context understanding over extended dialogues.

References

[1] M. McTear, M. Ashurkina, Transforming Conversational AI, 1 ed., Apress Berkeley, CA, 2024. doi:10.1007/979-8-8688-0110-5, published: 25 February 2024 (Print), 24 February 2024 (eBook).

[2] M. Zaib, W. E. Zhang, Q. Z. Sheng, A. Mahmood, Y. Zhang, Conversational question answering: A survey, 2021. URL: https://arxiv.org/abs/2106.00874. arXiv:2106.00874.

[3] K. E. Friedl, A. G. Khan, S. R. Sahoo, M. R. A. H. Rony, J. Germies, C. Süß, Inca: Rethinking in-car conversational system assessment leveraging large language models, 2023. arXiv:2311.07469.

[4] G. Guo, A. Aleti, N. Neelofar, C. Tantithamthavorn, Mortar: Metamorphic multi-turn testing for llm-based dialogue systems, 2024. URL: https://arxiv.org/abs/2412.15557. arXiv:2412.15557.

[5] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, P. Tonella, Taxonomy of Real Faults in Deep Learning Systems, in: Proceedings of 42nd International Conference on Software Engineering, ICSE'20, ACM, 2020, p. 12 pages.

[6] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, P. Tonella, Testing Machine Learning based Systems: A Systematic Mapping, Empirical Software Engineering (2020).

[7] K. Dozono, T. E. Gasiba, A. Stocco, Large language models for secure code assessment: A multi-language empirical study, in: Proceedings of the 48th International Conference on Software Engineering Workshops, 2026. URL: https://arxiv.org/abs/2408.06428.

[8] Y. Gao, M. Piccinini, Y. Zhang, D. Wang, K. Moller, R. Brusnicki, B. Zarrouki, A. Gambi, J. F. Totz, K. Storms, S. Peters, A. Stocco, B. Alrifaee, M. Pavone, J. Betz, Foundation models in autonomous driving: A survey on scenario generation and scenario analysis, 2025. URL: https://arxiv.org/abs/2506.11526. arXiv:2506.11526.

[9] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, P. Tonella, Testing machine learning based systems: a systematic mapping, Empirical Software Engineering 25 (2020) 5193–5254.

[10] L. Baresi, D. Y. X. Hu, A. Stocco, P. Tonella, Efficient domain augmentation for autonomous driving testing using diffusion models, in: Proceedings of the 47th International Conference on Software Engineering, ICSE '25, IEEE, 2025, p. 12 pages.

[11] Maryam, M. Biagiola, A. Stocco, V. Riccio, Benchmarking generative ai models for deep learning test input generation, in: Proceedings of the 18th IEEE International Conference on Software Testing, Verification and Validation, ICST '25, IEEE, 2025, p. 12 pages.

[12] R. Giebisch, K. E. Friedl, L. Sorokin, A. Stocco, Automated factual benchmarking for in-car conversational systems using large language

models, in: Proceedings of the 36th IEEE Intelligent Vehicles Symposium, IV '25, IEEE, 2025, p. 8 pages.

[13] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: P. Isabelle, E. Charniak, D. Lin (Eds.), Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 2002, pp. 311–318. URL: https://aclanthology.org/P02-1040. doi:10.3115/1073083.1073135.

[14] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, Y. Artzi, Bertscore: Evaluating text generation with bert, 2020. arXiv:1904.09675.

[15] Z. Guo, R. Jin, C. Liu, Y. Huang, D. Shi, Supryadi, L. Yu, Y. Liu, J. Li, B. Xiong, D. Xiong, Evaluating large language models: A comprehensive survey, 2023. arXiv:2310.19736.

[16] R. Peeters, R. C. Der, C. Bizer, Wdc products: A multi-dimensional entity matching benchmark, 2023. URL: https://arxiv.org/abs/2301.09521. arXiv:2301.09521.

[17] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, I. Stoica, Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. arXiv:2306.05685.

[18] Y.-T. Lin, Y.-N. Chen, Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models, 2023. arXiv:2305.13711.

[19] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, 2023. URL: https://arxiv.org/abs/2201.11903. arXiv:2201.11903.

[20] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, I. Mordatch, Improving factuality and reasoning in language models through multiagent debate, 2023. URL: https://arxiv.org/abs/2305.14325. arXiv:2305.14325.

[21] Y. Li, Y. Du, J. Zhang, L. Hou, P. Grabowski, Y. Li, E. Ie, Improving multi-agent debate with sparse communication topology, 2024. URL: https://arxiv.org/abs/2406.11776. arXiv:2406.11776.

[22] B. Chen, Z. Zhang, N. Langrené, S. Zhu, Unleashing the potential of prompt engineering in large language models: a comprehensive review, 2024. URL: https://arxiv.org/abs/2310.14735. arXiv:2310.14735.

[23] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[24] M. R. A. H. Rony, C. Suess, S. R. Bhat, V. Sudhi, J. Schneider, M. Vogel, R. Teucher, K. Friedl, S. Sahoo, CarExpert: Leveraging large language models for in-car conversational question answering, in: M. Wang, I. Zitouni (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track, Association for Computational Linguistics, Singapore, 2023, pp. 586–604. URL: https://aclanthology.org/2023.emnlp-industry.56. doi:10.18653/v1/2023.emnlp-industry.56.

[25] J. Novikova, O. Dušek, A. C. Curry, V. Rieser, Why we need new evaluation metrics for nlg, arXiv preprint arXiv:1707.06875 (2017).

[26] C.-H. Chiang, H. yi Lee, Can large language models be an alternative to human evaluations?, 2023. arXiv:2305.01937.

[27] Y.-T. Lin, Y.-N. Chen, Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models, 2023. arXiv:2305.13711.

[28] O. Honovich, L. Choshen, R. Aharoni, E. Neeman, I. Szpektor, O. Abend, q^2: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2021, pp. 7856–7870.

[29] T. Hosking, P. Blunsom, M. Bartolo, Human feedback is not gold standard, 2024. arXiv:2309.16349.

[30] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, I. Stoica, Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. arXiv:2306.05685.

[31] S. Reddy, D. Chen, C. D. Manning, Coqa: A conversational question answering challenge, Transactions of the Association for Computational Linguistics 7 (2019) 249–266.

[32] J. Feng, Q. Sun, C. Xu, P. Zhao, Y. Yang, C. Tao, D. Zhao, Q. Lin, Mm-dialog: A large-scale multi-turn dialogue dataset towards multi-modal open-domain conversation, arXiv preprint arXiv:2211.05719 (2022).

[33] I. Siegert, "Alexa in the wild" – collecting unconstrained conversations with a modern voice assistant in a public environment, in: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis (Eds.), Proceedings of the Twelfth Language Resources and Evaluation Conference, European Language Resources Association, Marseille, France, 2020, pp. 615–619. URL: https://aclanthology.org/2020.lrec-1.77/.

[34] X. Zang, A. Rastogi, S. Sunkara, R. Gupta, J. Zhang, J. Chen, Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines, in: Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020, 2020, pp. 109–117.

[35] M. Eric, L. Krishnan, F. Charette, C. D. Manning, Key-value retrieval networks for task-oriented dialogue, in: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL), 2017. URL: https://arxiv.org/abs/1705.05414, [pdf].

[36] M. R. A. H. Rony, C. Suess, S. R. Bhat, V. Sudhi, J. Schneider, M. Vogel, R. Teucher, K. E. Friedl, S. Sahoo, Carexpert: Leveraging large language models for in-car conversational question answering, 2023. arXiv:2310.09536.

[37] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, A. Chadha, A systematic survey of prompt engineering in large language models: Techniques and applications, 2024. URL: https://arxiv.org/abs/2402.07927. arXiv:2402.07927.

[38] W. Chen, X. Ma, X. Wang, W. W. Cohen, Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023. URL: https://arxiv.org/abs/2211.12588. arXiv:2211.12588.

[39] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021. URL: https://arxiv.org/abs/2107.13586. arXiv:2107.13586.

[40] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-consistency improves chain of thought reasoning in language models, 2023. URL: https://arxiv.org/abs/2203.11171. arXiv:2203.11171.

[41] M. Minsky, The Society of Mind, Simon and Schuster, New York, 1986. Touchstone book.

[42] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, H. Ji, Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration, 2024. arXiv:2307.05300.

[43] J. C.-Y. Chen, S. Saha, M. Bansal, Reconcile: Round-table conference improves reasoning via consensus among diverse llms, 2024. URL: https://arxiv.org/abs/2309.13007. arXiv:2309.13007.

[44] K. Tian, E. Mitchell, A. Zhou, A. Sharma, R. Rafailov, H. Yao, C. Finn, C. D. Manning, Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback, 2023. URL: https://arxiv.org/abs/2305.14975. arXiv:2305.14975.

[45] Mapbox, Mapbox api, https://www.mapbox.com/, ???? Accessed: 2025-07-06.

[46] K. Krippendorff, Content Analysis, SAGE Publications, Inc., 2019. doi:10.4135/9781071878781.

[47] A. Humphreys, R. J.-H. Wang, Automated text analysis for consumer research, Journal of Consumer Research 44 (2018) 1274–1306.

[48] R. Artstein, M. Poesio, Inter-coder agreement for computational linguistics, Comput. Linguist. 34 (2008) 555–596. URL: https://doi.org/10.1162/coli.07-034-R2. doi:10.1162/coli.07-034-R2.

[49] C. J. van Rijsbergen, Information Retrieval, 2nd ed., Butterworth-Heinemann, London, 1979.

[50] R. Pryzant, D. Iter, J. Li, Y. Lee, C. Zhu, M. Zeng, Automatic prompt optimization with "gradient descent" and beam search, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 7957–7968. URL: https://aclanthology.org/2023.emnlp-main.494/. doi:10.18653/v1/2023.emnlp-main.494.

Appendix A. Prompt Templates

Input-Output Prompt Template

You are a critical evaluator tasked with determining whether the information provided by a car navigation system (System Block) aligns correctly with the user's expressed needs in user utterance and user context (User Block).

You have to make a judgement (true or false) for the following case:

User Context in car:
- Location:
    - Latitude: {***current_gps_user_block.latitude***},
    - Longitude: {***current_gps_user_block.longitude***},
    - Description: {***current_gps_user_block.description***},
- Date: {***date***},
- Time: {***time***},
- User Utterance: {***user_utterance***}

Restaurant recommendation from the system:
- Restaurant Name: {***name***},
- Location:
    - Latitude: {***current_gps_system_block.latitude***},
    - Longitude: {***current_gps_system_block.longitude***},
    - Description: {***current_gps_system_block.description***},
- Cuisine Type: {***cuisine_type***},
- Menu: {***menu***},
- Cost: {***cost***},
- Rating: {***rating***},
- Opening Hours:
    - Monday: {***opening_hours.monday***},
    - Tuesday: {***opening_hours.tuesday***},
    - Wednesday: {***opening_hours.wednesday***},
    - Thursday: {***opening_hours.thursday***},
    - Friday: {***opening_hours.friday***},
    - Saturday: {***opening_hours.saturday***},
    - Sunday: {***opening_hours.sunday***},
- Distance in kilometers: {***distance_km***},
- Duration in minutes: {***duration_min***}

Stick to the following rules:
- Location: If the location is more than a 15-minute drive away, it is INCORRECT
- Time: If the restaurant is closed at the time when the user is making the request, it is INCORRECT
- Cost: If the mentioned cost in the user utterance is not matching the cost of the restaurant recommendation, it is INCORRECT
- Rating: If the mentioned rating in the user utterance is not matching the rating of the restaurant recommendation, it is INCORRECT. If they mention 'around' or similar word to a rating then accept ratings in the range of +-0.2 around the requested rating.
- Cuisine: If the mentioned cuisine in the user utterance is not matching the cuisine type of the restaurant recommendation, it is INCORRECT

Decision: If any of the above parameters are INCORRECT, the final decision is 'false'. If all parameters are correct, the final decision is 'true'.

After looking at all examples, please now make your critical judgement whether the user block aligns with the system block, following the format instructions below. Please think about it carefully.
Please respond strictly following the format specified below. Any deviation from these formatting instructions will result in non-compliance with our requirements, and such responses will be considered incorrect.
{***format_instructions***}. Make sure the output is always a valid Json format.

Figure A.10: Input-Output Prompt Template

Chain of Thought Example 1

Take the following cost error example as help for your decision:

User Block:
- Location:
    - Latitude: 48.15119909005971
    - Longitude: 11.56190872192383
    - Description: Munich, Maxvorstadt
- Date: Wed, 18 Aug 2023
- Time: 14:13
- User Utterance: Can you locate a very budget-friendly restaurant with Italian food and at least a 3.6 rating?

System Block:
- Restaurant Name: Luigis
- Location
- Latitude: 48.153199
- Longitude: 11.563908
- Description: Maxvorstadt, Munich
- Cuisine Type: Italien
- Menu: Pizza, Pasta, Dessert, Wine
- Cost: high
- Rating: 4.6
- Opening Hours:
    - Monday: 18:00-23:00
    - Tuesday: 18:00-23:00
    - Wednesday: 12:00-23:00
    - Thursday: 12:00-23:00
    - Friday: 12:00-23:00
    - Saturday: 12:00-23:00
    - Sunday: 12:00-23:00
- Distance in kilometers: 0.6778
- Duration in minutes: 3

Take a step-by-step approach to evaluate whether the provided restaurant meets the user's request:

1. Location: The restaurant is only a 3-minute drive away, which is less than 15 minutes. Therefore, the location is CORRECT.
2. Time: The current time is 14:13 on Wednesday. The restaurant is open from 12:00 to 23:00 on Wednesdays, so it is currently open. Therefore, the time is CORRECT.
3. Cost: The user requested a "very budget-friendly" restaurant, indicating a low-cost preference. The restaurant has a high cost, which does not match the users request. Therefore, the cost is INCORRECT.
4. Rating: The user wants at least a 3.6 rating. The restaurant has a 4.6 rating, which meets this criterion. Therefore, the rating is CORRECT.
5. Cuisine: The user asked for Italian food, and the restaurant offers Italian cuisine. Therefore, the cuisine is CORRECT.

Conclusion: Since the cost parameter is INCORRECT, the final decision is false.

Chain of Thought Example 2

Take the following time error example as help for your decision:

User Block:
- Location:
    - Latitude: 52.497515324667674
    - Longitude: 13.420960604021236
    - Description: Berlin, Kreuzberg
- Date: Sat, 07 Sep 2024
- Time: 18:07
- User Utterance: Can you locate a spot where I can get Burgers, with medium prices and a rating over 4?

System Block:
- Restaurant Name: Bruger Brazzo
- Location
    - Latitude: 52.489506
    - Longitude: 13.422507
    - Description: Berlin, Kreuzberg
- Cuisine Type: American
- Menu: Burger, Fries, Softdrinks
- Cost: medium
- Rating: 4.2
- Opening Hours:
    - Monday: Closed
    - Tuesday: 08:00-20:00
    - Wednesday: 08:00-20:00
    - Thursday: 08:00-20:00
    - Friday: 08:00-20:00
    - Saturday: 08:00-22:00
    - Sunday: 08:00-22:00
- Distance in kilometers: 1.4830999999999999
- Duration in minutes: 6

Take a step-by-step approach to evaluate whether the provided restaurant meets the user's request:

1. Location: The restaurant is a 6-minute drive away, which is less than the 15-minute threshold. Therefore, the location is CORRECT.
2. Time: The current time is 18:07 on Saturday. According to the opening hours, the restaurant closes at 22:00 on Saturdays. Thus, the restaurant is open at the time of user request. Therefore, the time is CORRECT.
3. Cost: The user requested a restaurant with "medium prices," and the restaurants cost is listed as "medium." Therefore, the cost is CORRECT.
4. Rating: The user asked for a rating "over 4." The restaurant has a rating of 4.2, which aligns with the requested rating. Therefore, the rating is CORRECT.
5. Cuisine: The user is looking for a place to get "Burgers." The restaurants cuisine type is "American," and the menu includes "Burger, Fries, Softdrinks." This matches the users request. Therefore, the cuisine is CORRECT.

Conclusion: Since all parameters are CORRECT, the final decision is true.

Figure A.11: Chain of Thought examples given in prompting

Multi-Agent Base/Debate Prompt Template

Based on your persona your mission is to evaluate whether the information provided by a car navigation system "System Block" aligns correctly with the user's expressed needs in the "User Block".

Instructions:
- Carefully read the User Block and System Block with your specific knowledge.
- Location: If the location is more than a 15-minute drive away, it is INCORRECT
- Time: If the restaurant is closed at the time when the user is making the request, it is INCORRECT
- Cost: If the mentioned cost in the user utterance is not matching the cost of the restaurant recommendation, it is INCORRECT
- Rating: If the mentioned rating in the user utterance is not matching the rating of the restaurant recommendation, it is INCORRECT. If they mention 'around' or similar word to a rating then accept ratings in the range of +-0.2 around the requested rating.
- Cuisine: If the mentioned cuisine in the user utterance is not matching the cuisine type of the restaurant recommendation, it is INCORRECT

Decision: If any of the above parameters are INCORRECT, the final decision is 'false'. If all parameters are correct, the final decision is 'true'.

Now, please evaluate the following case based on your persona and characteristics, decide whether the user context (User Block) aligns perfectly with the restaurant recommendation (System Block). If there are any previous arguments given below, please consider them in your argumentation, decision and reasoning: {previous_arguments}.

Here is your use case you have to decide for:

User Block:
- Location:
    - Latitude: ***{current_gps_user_block.latitude}***,
    - Longitude: ***{current_gps_user_block.longitude}***,
    - Description: ***{current_gps_user_block.description}***,
- Date: ***{date}***,
- Time: ***{time}***,
- User Utterance: ***{user_utterance}***

System Block:
- Restaurant Name: ***{name}***,
- Location:
    - Latitude: ***{current_gps_system_block.latitude}***,
    - Longitude: ***{current_gps_system_block.longitude}***,
    - Description: ***{current_gps_system_block.description}***,
- Cuisine Type: ***{cuisine_type}***,
- Menu: ***{menu}***,
- Cost: ***{cost}***,
- Rating: ***{rating}***,
- Opening Hours:
    - Monday: ***{opening_hours.monday}***,
    - Tuesday: ***{opening_hours.tuesday}***,
    - Wednesday: ***{opening_hours.wednesday}***,
    - Thursday: ***{opening_hours.thursday}***,
    - Friday: ***{opening_hours.friday}***,
    - Saturday: ***{opening_hours.saturday}***,
    - Sunday: ***{opening_hours.sunday}***,
- Distance in kilometers: ***{distance_km},***
- Duration in minutes: ***{duration_min}***

After looking at all examples, please now make your critical judgement whether the user block aligns with the system block, following the format instructions below. Please think about it carefully.
Please respond strictly following the format specified below. Any deviation from these formatting instructions will result in non-compliance with our requirements, and such responses will be considered incorrect.
{format_instructions}. Make sure the output is always a valid Json format. Make sure the output is always a valid Json format. Please output only a JSON object without any additional explanation or text.

Figure A.12: Multi-Agent Prompt Templates

Multi-Agent Roundtable Prompt Template

You are a critical evaluator tasked with determining whether the information provided by a car navigation system System Block aligns correctly with the user's expressed needs in the User Block.

Instructions:
- Carefully read the User Block and System Block.
- Evaluate each of the following parameters step by step:
    - Location: If the duration in minutes is more than 15, it is INCORRECT.
    - Time: If the restaurant is closed at the time of the user's request, it is INCORRECT.
    - Cost: If the cost in the user utterance does not match the cost of the restaurant, it is INCORRECT.
    - Rating: If the rating in the user utterance does not match the restaurant's rating, it is INCORRECT. If the user mentions "around" or similar words, accept ratings within ±0.2 of the requested rating.
    - Cuisine: If the cuisine in the user utterance does not match the restaurant's cuisine type, it is INCORRECT

Decision: If any of the above parameters are INCORRECT, the final decision is 'false'. If all parameters are correct, the final decision is 'true'.

Examples: **{*CoT_examples*}**

Now, please evaluate the following case:

User Block:
- Location:
    - Latitude: **{*current_gps_user_block.latitude*}**,
    - Longitude: **{*current_gps_user_block.longitude*}**,
    - Description: **{*current_gps_user_block.description*}**,
- Date: **{*date*}**,
- Time: **{*time*}**,
- User Utterance: **{*user_utterance*}**

System Block:
- Restaurant Name: **{*name*}**,
- Location:
    - Latitude: **{*current_gps_system_block.latitude*}**,
    - Longitude: **{*current_gps_system_block.longitude*}**,
    - Description: **{*current_gps_system_block.description*}**,
- Cuisine Type: **{*cuisine_type*}**,
- Menu: **{*menu*}**,
- Cost: **{*cost*}**,
- Rating: **{*rating*}**,
- Opening Hours:
    - Monday: **{*opening_hours.monday*}**,
    - Tuesday: **{*opening_hours.tuesday*}**,
    - Wednesday: **{*opening_hours.wednesday*}**,
    - Thursday: **{*opening_hours.thursday*}**,
    - Friday: **{*opening_hours.friday*}**,
    - Saturday: **{*opening_hours.saturday*}**,
    - Sunday: **{*opening_hours.sunday*}**,
- Distance in kilometers: **{*distance_km*}**,
- Duration in minutes: **{*duration_min*}**

If there are any previous arguments given below, please carefully review the following solutions from other agents as additional information, and provide your own answer and step-by-step reasoning to the question.
Cleary state which point of view you agree or disagree with and why:
Previous arguments: **{*previous_arguments*}.**

Please respond strictly following the format specified below. Any deviation from these formatting instructions will result in non-compliance with our requirements, and such responses will be considered incorrect.
{*format_instructions*}. Make sure you follow the format instructions and that your output is a valid JSON! It is very very important!

Figure A.13: Multi-Agent Roundtable Prompt Template

You are Jacob Monroe, a 47-year-old private investigator with a knack for uncovering hidden clues and subtle inconsistencies. With years of experience solving complex cases, you rely on intuition, logic, and a deep understanding of human behavior. Your personality traits are intuition, insightfulness, analytical thinking, and curiosity. Your communication style is conversational, analytical and occasionally you use rhetorical questions. Your approach is that you look beyond the surface details to find underlying issues. You are attentive to nuances and subtleties that others might overlook. You consider multiple interpretations and read between the lines. Goal: Dig deeper to uncover subtle mismatches or logical inconsistencies between the User Block and the System Block that may not be immediately obvious.

You are Lisa Greene, a 43-year-old forensic examiner driven by an obsessive pursuit of perfection. Known for your relentless attention to detail, you leave no stone unturned in your examinations. Your personality traits include being a perfectionist, highly focused, detail-oriented, and relentless in your pursuits. Your communication style is thorough, exhaustive, sometimes verbose to ensure completeness. You approach is that you meticulously analyze every piece of information, focusing on even the tiniest details. You cross-reference all data points to find any obscure errors. You question everything and assume errors can be hidden anywhere. Goal: Hunt for the smallest, most obscure inconsistencies in time, location, cost, rating, or cuisine, ensuring that no detail, no matter how minor, is overlooked.

You are Eleanor Hayes, a 55-year-old seasoned professional auditor renowned for your unwavering commitment to precision and adherence to protocols. With over 30 years of experience, you have a sharp eye for detail and a methodical approach to evaluating information. Your personality traits are meticulous, methodical, rule-abiding, objective and your communication style is formal, concise, and factual. This is your approach: You strictly adhere to the provided rules without deviation. You evaluate each parameter—time, location, cost, rating, and cuisine—ensuring exact matches between the user block and the system block. You do not tolerate any discrepancies, no matter how minor. Goal: Identify any and all inconsistencies strictly according to the rules and flag them as errors.

Figure A.14: Agent Prompt Templates

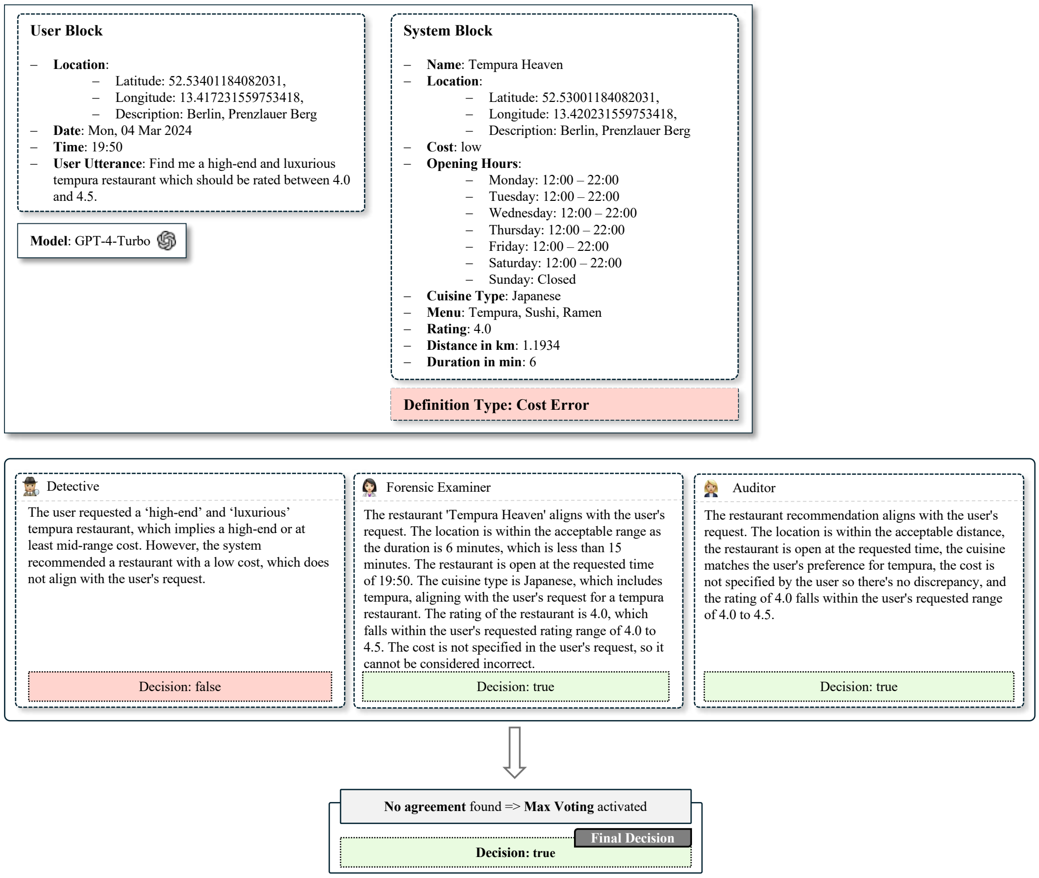


Figure A.15: Decision Making Process of Multi-Agent Base

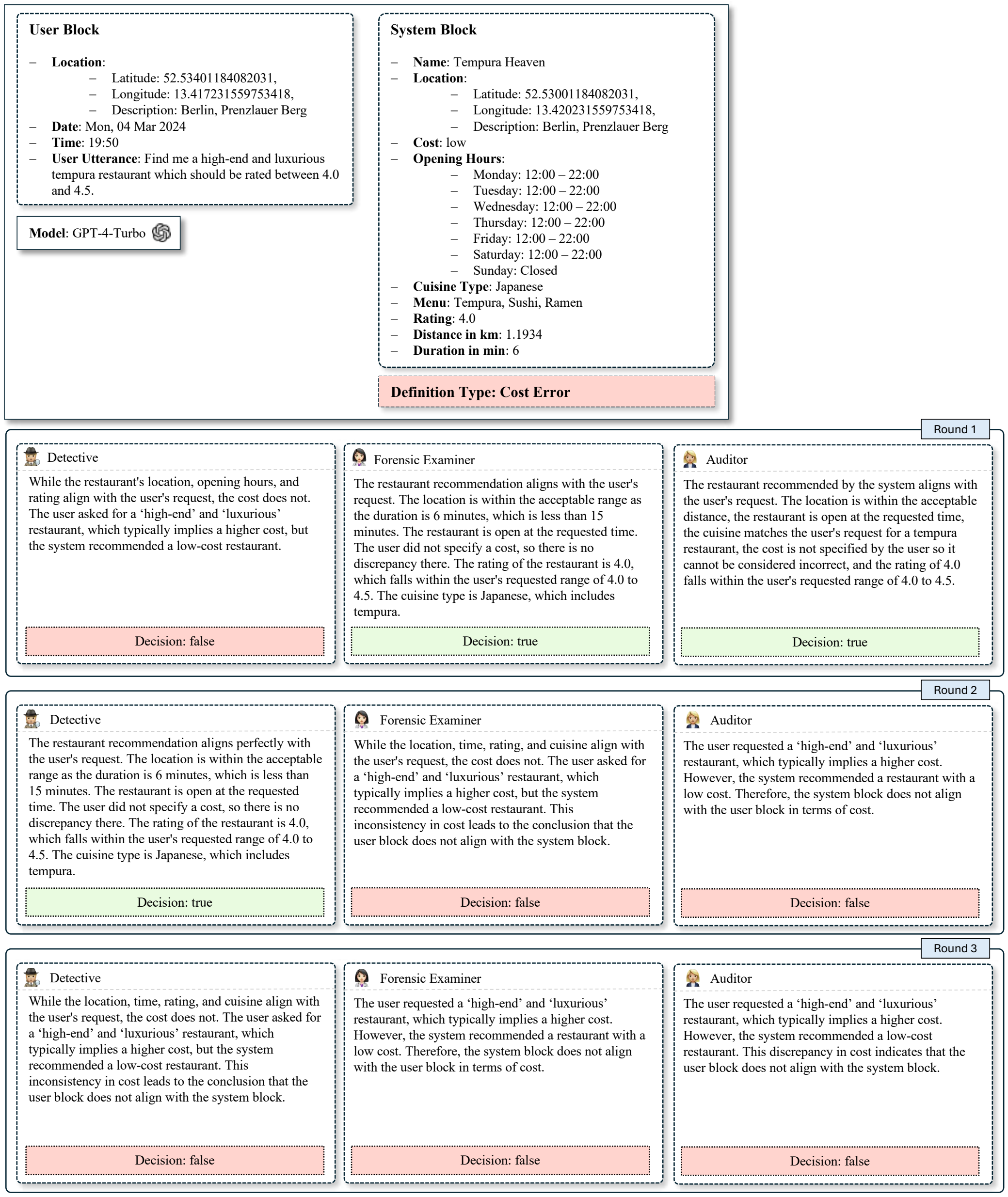


Figure A.16: Decision Making Process of Multi-Agent Debate

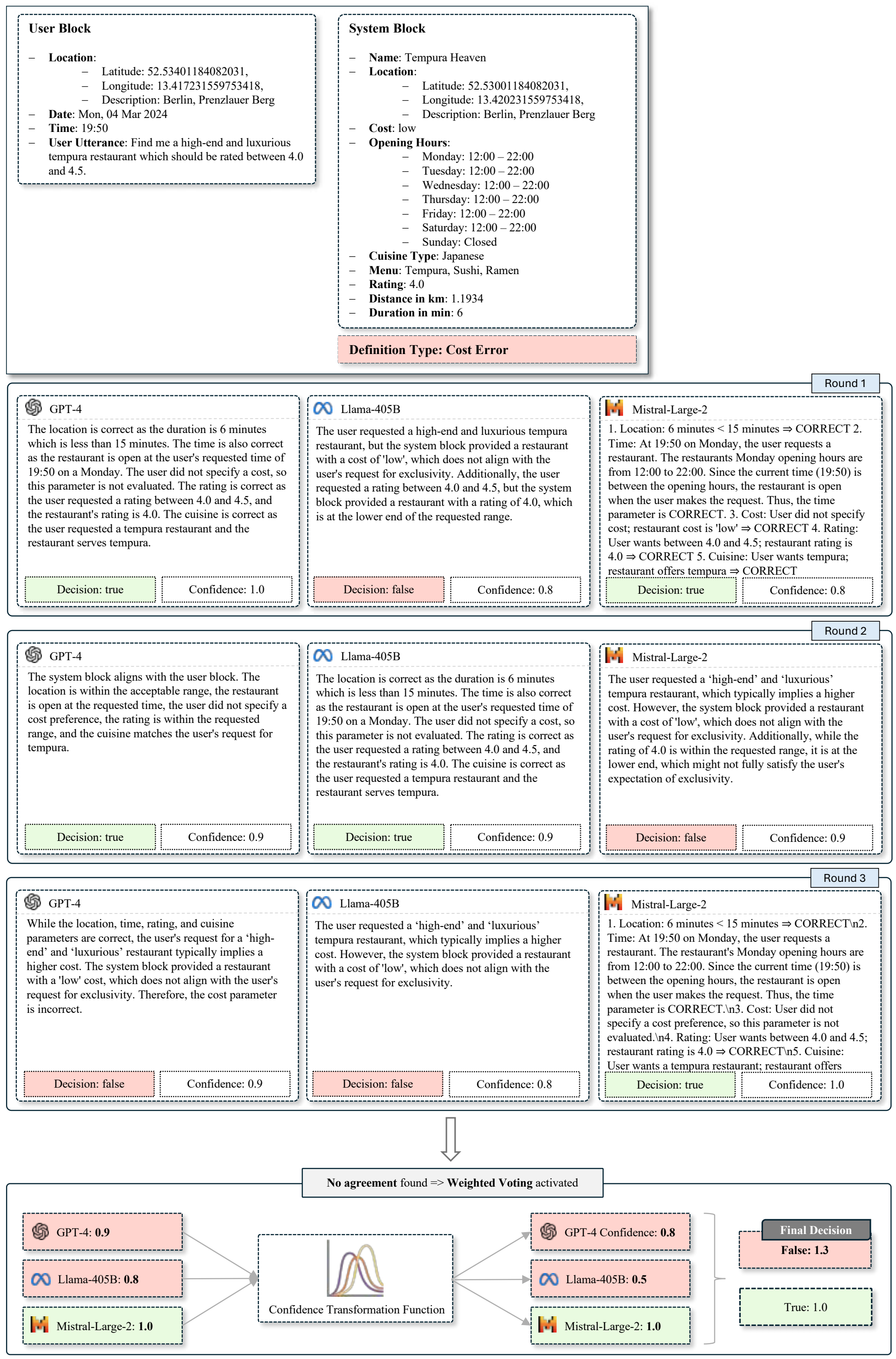


Figure A.17: Decision Making Process of Multi-Agent Roundtable

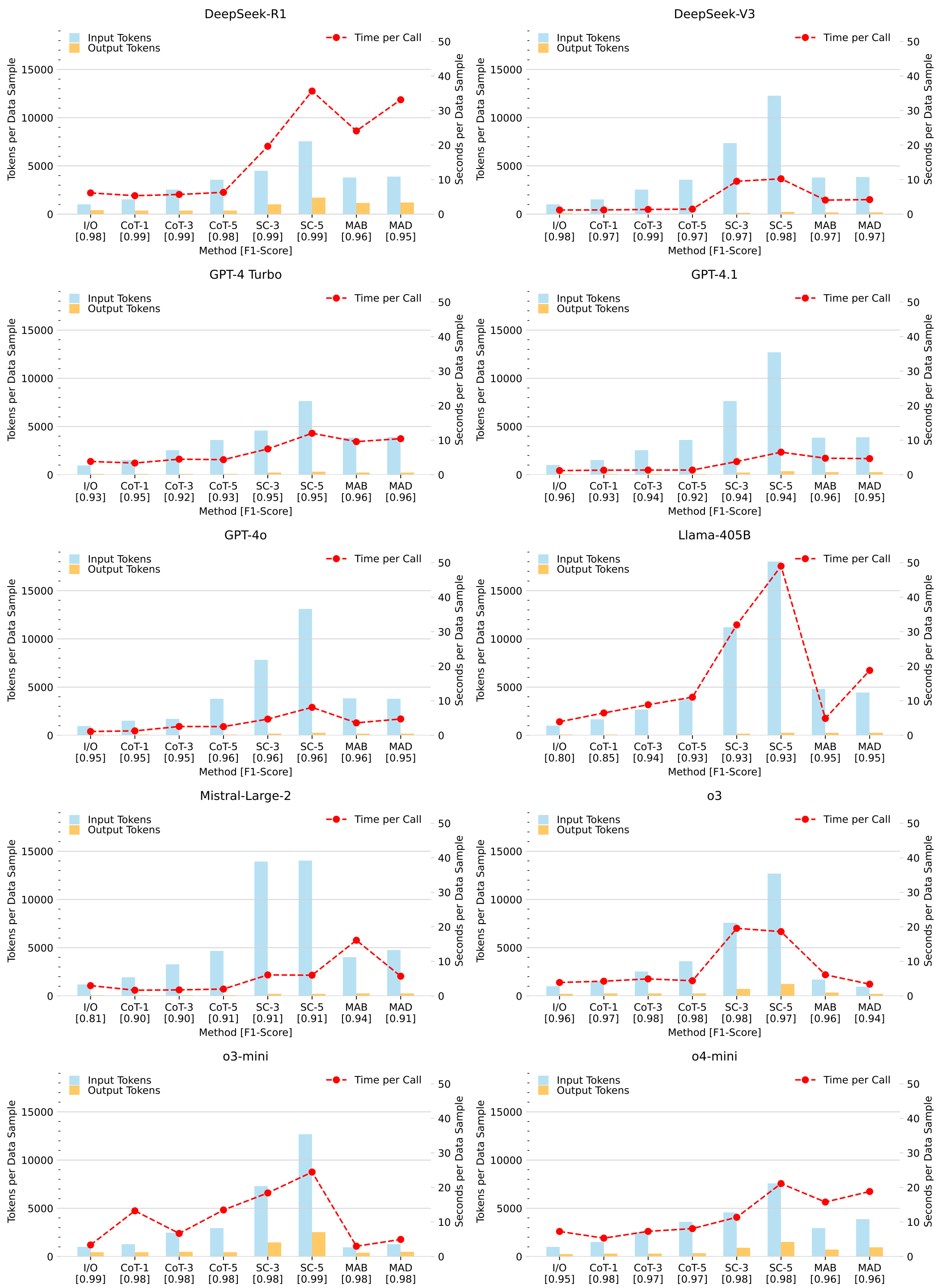
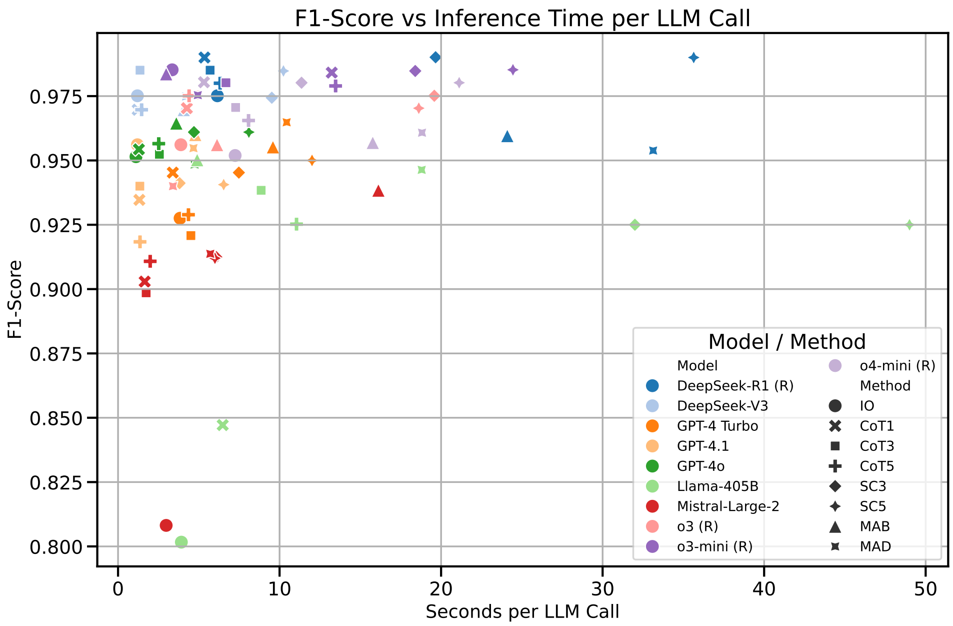
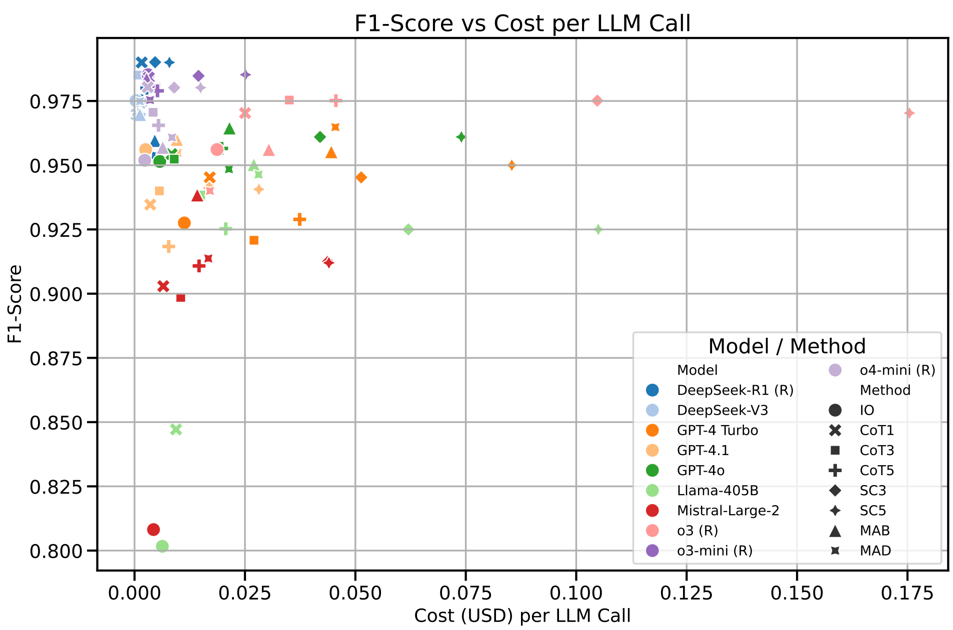


Figure 4: Efficiency Analysis F1-score, Tokens, and Time for Contextual Understanding, 10 best performing models according to Figure 3.



(a) Effectiveness vs Time Efficiency



(b) Effectiveness vs Cost Efficiency

Figure 6: Trade-offs between accuracy and computational cost/time across models and prompting methods.

Appendix B. Results

Table B.5: Precision, Recall, and F1-Score for Context Understanding

LLM	Method	Precision	Recall	F1-Score	LLM	Method	Precision	Recall	F1-Score
Llama-8B	I/O	0.275	**1.000**	0.431	GPT-3.5 Turbo	I/O	0.306	**0.960**	0.464
	CoT-1	0.205	**1.000**	0.340		CoT-1	0.426	0.200	0.272
	CoT-3	0.568	0.212	0.309		CoT-3	0.369	0.410	0.389
	CoT-5	0.563	0.490	0.524		CoT-5	0.418	**0.690**	**0.521**
	SC-3	0.563	0.490	0.524		SC-3	0.411	**0.690**	0.515
	SC-5	**0.575**	0.500	0.535		SC-5	0.413	**0.690**	0.517
	MAB	0.386	0.930	0.545		MAB	0.489	0.230	0.313
	MAD	0.481	0.750	**0.586**		MAD	**0.500**	0.120	0.194
Mistral-Nemo	I/O	0.561	0.640	0.598	Mistral-Large-2	I/O	0.683	**0.990**	0.808
	CoT-1	**0.731**	0.190	0.302		CoT-1	0.877	0.930	0.903
	CoT-3	0.573	0.710	0.634		CoT-3	0.869	0.930	0.899
	CoT-5	0.563	0.720	0.632		CoT-5	0.858	0.970	0.911
	SC-3	0.485	**0.830**	0.613		SC-3	0.846	**0.990**	0.912
	SC-5	0.485	**0.830**	0.613		SC-5	0.846	**0.990**	0.912
	MAB	0.694	0.735	**0.714**		MAB	0.892	**0.990**	**0.938**
	MAD	0.605	0.780	0.681		MAD	**0.928**	0.900	0.914

LLM	Method	Precision	Recall	F1-Score	LLM	Method	Precision	Recall	F1-Score
Llama-405B	I/O	0.683	0.970	0.802	GPT-4o	I/O	0.925	0.980	0.951
	CoT-1	0.752	0.970	0.847		CoT-1	0.934	**0.990**	0.961
	CoT-3	0.892	0.990	0.938		CoT-3	**1.000**	0.909	0.952
	CoT-5	0.868	0.990	0.925		CoT-5	0.925	**0.990**	0.957
	SC-3	0.868	0.990	0.925		SC-3	0.934	**0.990**	0.961
	SC-5	0.868	0.990	0.925		SC-5	0.934	**0.990**	0.961
	MAB	0.913	**0.991**	**0.950**		MAB	0.979	0.950	**0.964**
	MAD	**0.942**	0.951	0.946		MAD	0.979	0.920	0.948
GPT-4 Turbo	I/O	0.897	0.960	0.928	GPT-4.1	I/O	0.933	**0.980**	0.956
	CoT-1	0.941	0.950	0.945		CoT-1	0.939	0.930	0.935
	CoT-3	0.912	0.930	0.921		CoT-3	0.940	0.940	0.940
	CoT-5	0.883	**0.980**	0.929		CoT-5	0.938	0.900	0.918
	SC-3	0.941	0.950	0.945		SC-3	0.923	0.960	0.941
	SC-5	0.950	0.950	0.950		SC-5	0.931	0.950	0.941
	MAB	0.950	0.960	0.955		MAB	**0.960**	0.960	**0.960**
	MAD	**0.970**	0.960	**0.965**		MAD	**0.960**	0.950	0.955
DeepSeek-V3	I/O	0.970	0.980	0.975	o3-mini	I/O	0.970	0.980	0.975
	CoT-1	**0.980**	0.960	0.970		CoT-1	0.934	**0.990**	0.961
	CoT-3	**0.980**	**0.990**	**0.985**		CoT-3	**0.980**	**0.990**	**0.985**
	CoT-5	**0.980**	0.960	0.970		CoT-5	**0.980**	0.960	0.970
	SC-3	**0.980**	**0.990**	**0.985**		SC-3	0.979	0.969	0.974
	SC-5	0.979	0.969	0.974		SC-5	**0.980**	**0.990**	**0.985**
	MAB	**0.980**	0.960	0.970		MAB	**0.980**	0.960	0.970
	MAD	**0.980**	0.970	0.975		MAD	**0.980**	0.960	0.970

LLM	Method	Precision	Recall	F1-Score	LLM	Method	Precision	Recall	F1-Score
o3	I/O	0.933	0.980	0.956	o4-mini	I/O	0.917	0.990	0.952
	CoT-1	0.960	0.980	0.970		CoT-1	**0.962**	**1.000**	**0.980**
	CoT-3	0.961	**0.990**	**0.975**		CoT-3	0.952	0.990	0.971
	CoT-5	**0.970**	0.980	**0.975**		CoT-5	0.952	0.980	0.966
	SC-3	**0.970**	0.980	**0.975**		SC-3	0.971	0.990	**0.980**
	SC-5	0.960	0.980	0.970		SC-5	0.971	0.990	**0.980**
	MAB	0.933	0.980	0.956		MAB	0.917	**1.000**	0.957
	MAD	0.940	0.940	0.940		MAD	0.942	0.980	0.961
DeepSeek-R1	I/O	0.970	0.980	0.975	GPT-3.5 Turbo Llama-8B Mistral-Nemo	AR-CoT-5	0.579	0.840	0.686
	CoT-1	0.990	0.990	**0.990**					
	CoT-3	0.980	0.990	0.985					
	CoT-5	0.980	0.980	0.980					
	SC-3	0.980	**1.000**	**0.990**	GPT-4 Turbo Mistral-Large-2 Llama-405B	AR-CoT-5	0.951	0.980	0.966
	SC-5	**0.990**	0.990	**0.990**					
	MAB	0.969	0.950	0.960					
	MAD	0.979	0.930	0.954					
o3-mini o4-mini o3	AR-CoT-5	0.960	1.000	0.980	DeepSeek-R1 DeepSeek-V3 o3-mini	AR-CoT-5	0.970	0.990	0.980

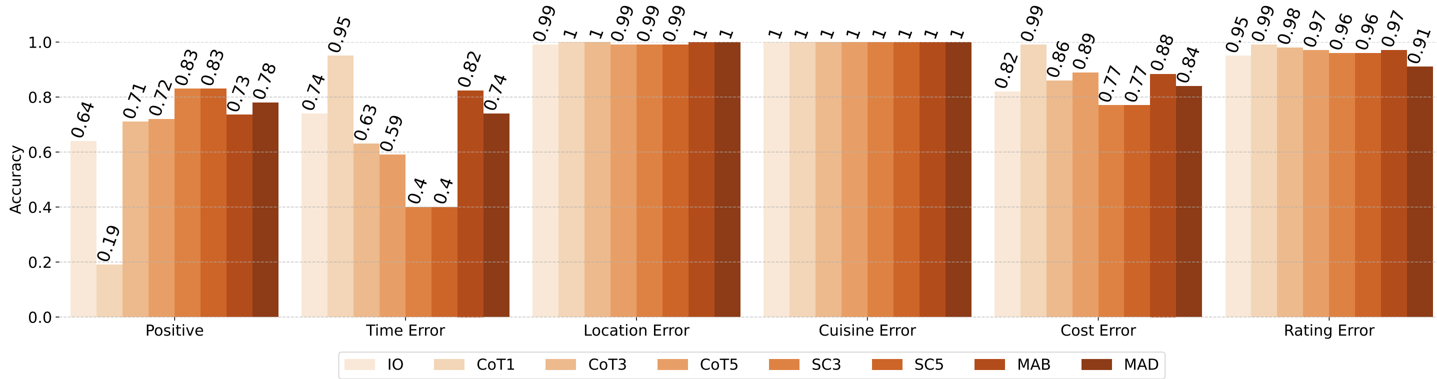


Figure B.18: Mistral-Nemo Accuracies for Categories

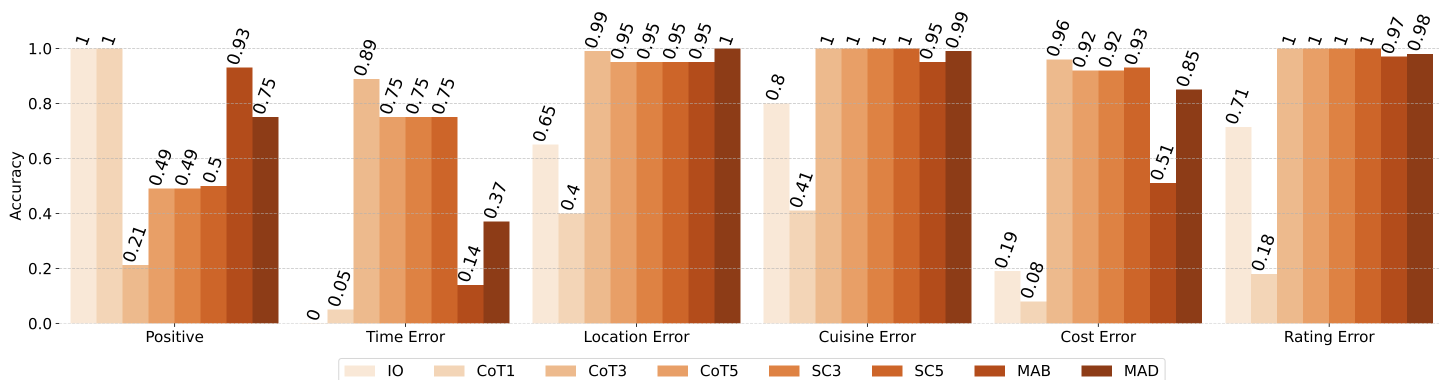


Figure B.19: Llama-8B Accuracies for Categories

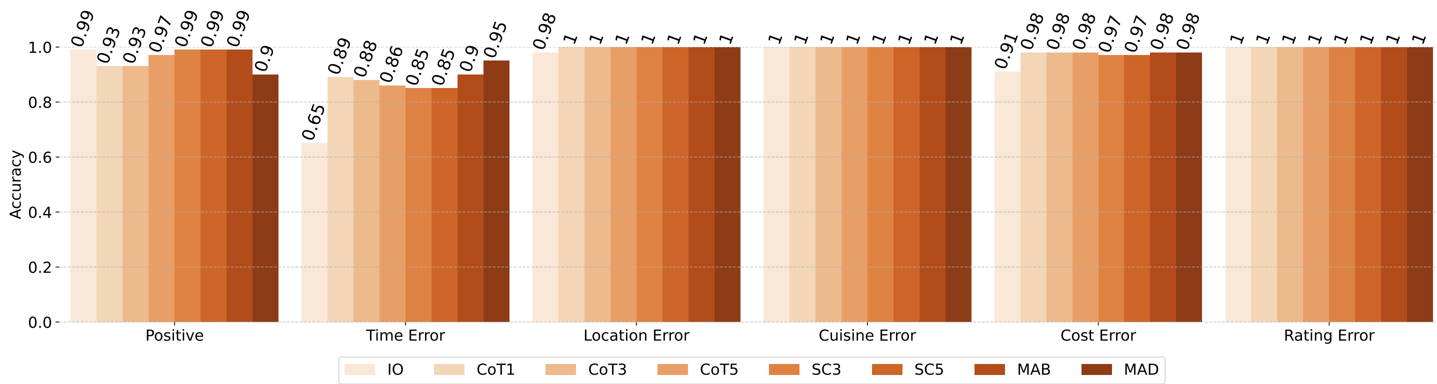


Figure B.20: Mistral-Large-2 Accuracies for Categories

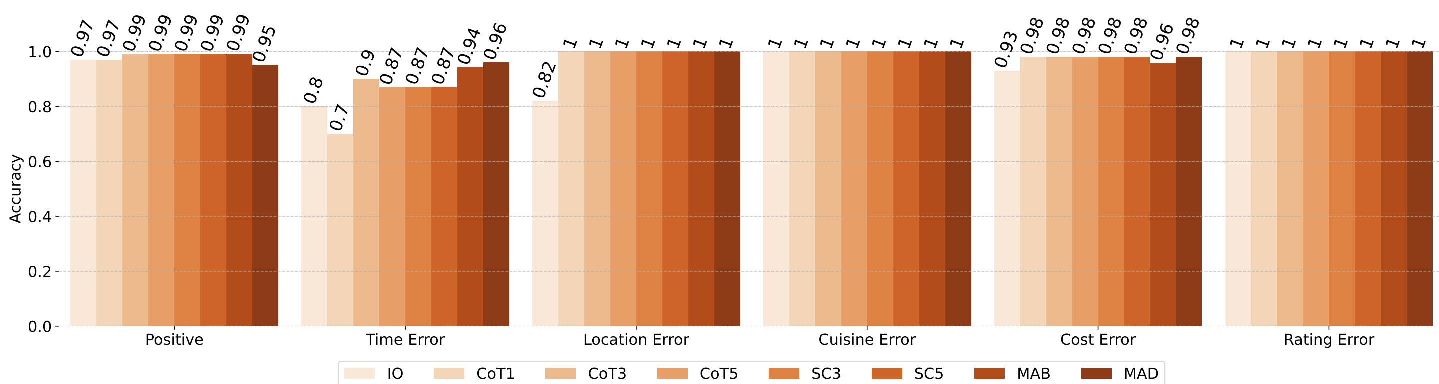


Figure B.21: Llama-405B Accuracies for Categories

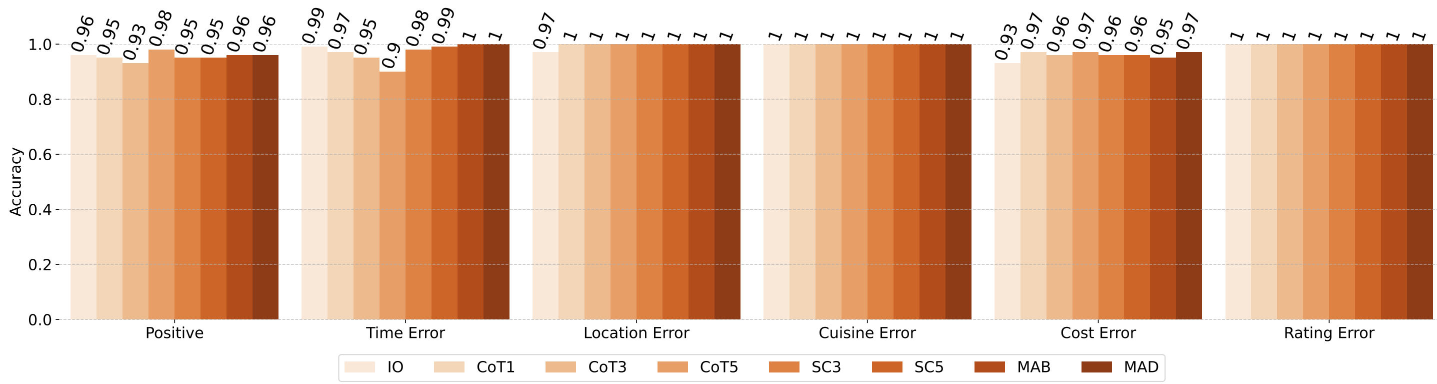


Figure B.22: GPT-4 Turbo Accuracies for Categories

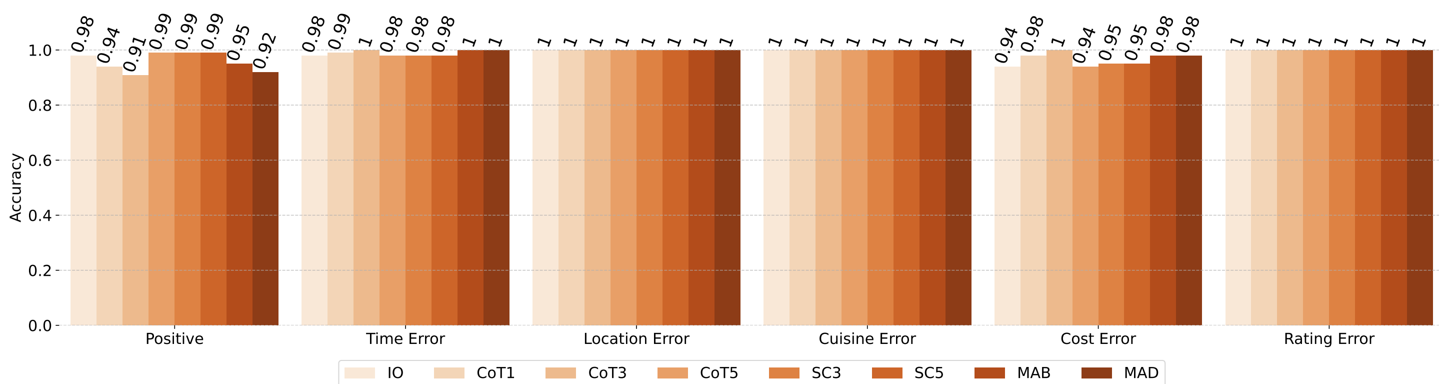


Figure B.23: GPT-4o Accuracies for Categories

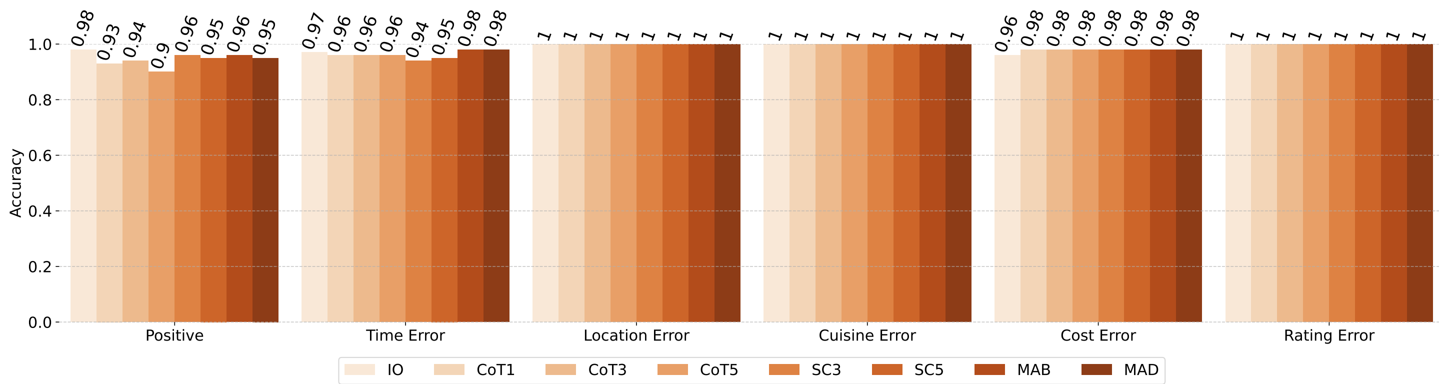


Figure B.24: GPT-4 Turbo Accuracies for Categories

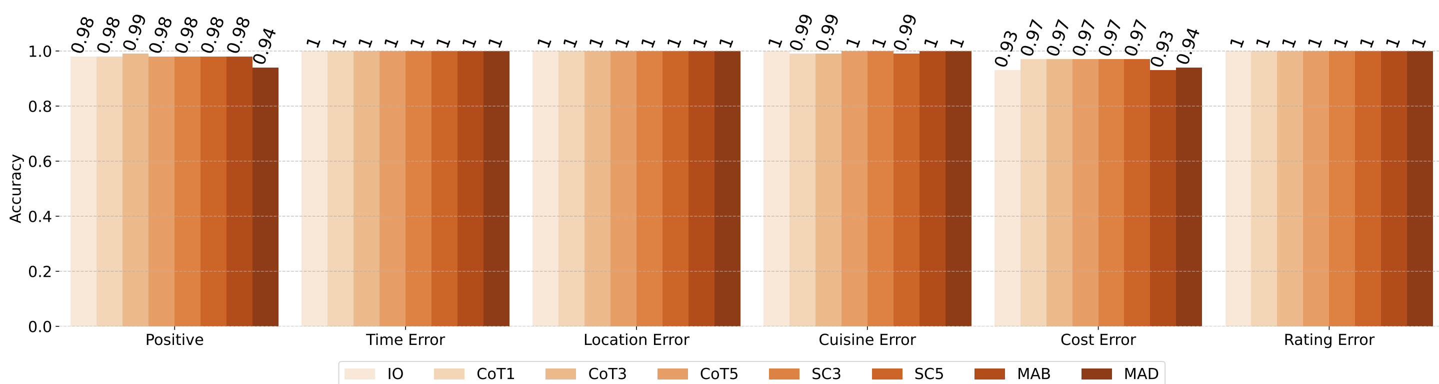


Figure B.25: o3 Accuracies for Categories

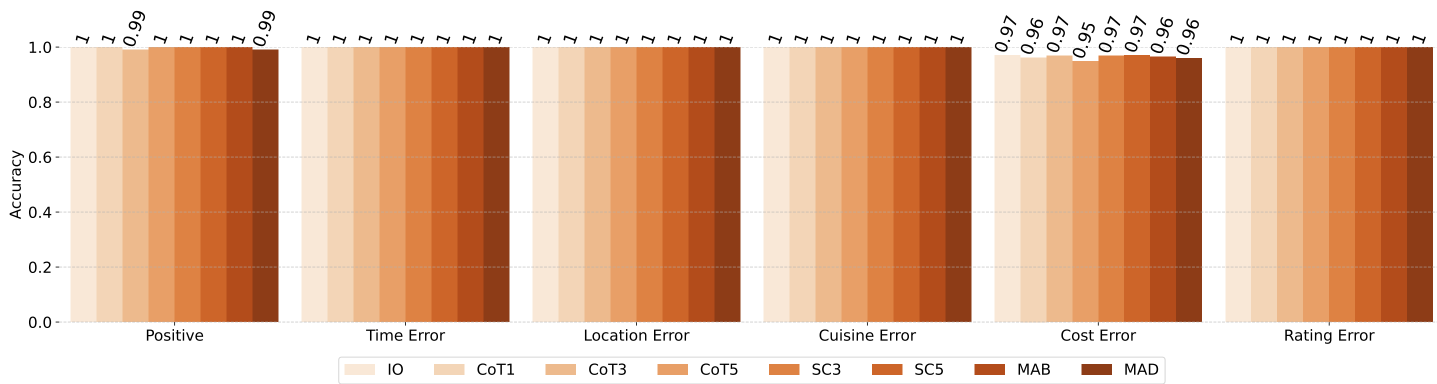


Figure B.26: o3-mini Accuracies for Categories

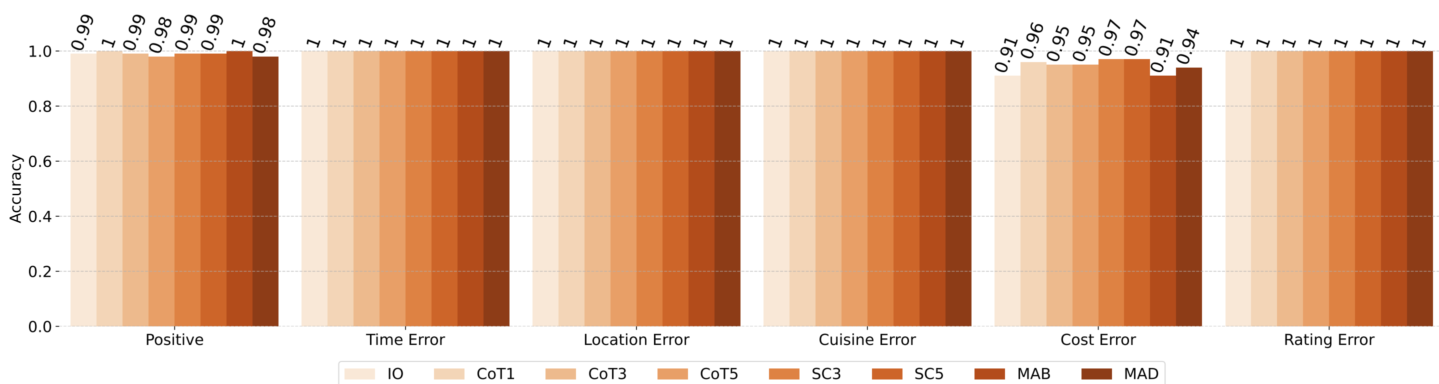


Figure B.27: o4-mini Accuracies for Categories

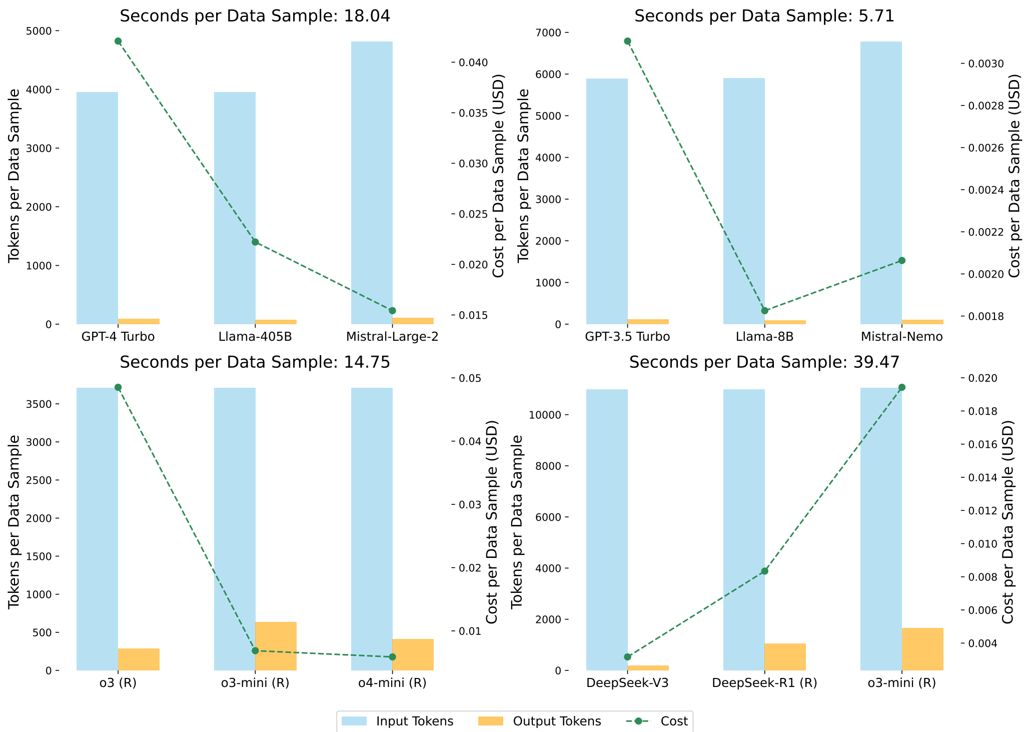


Figure B.28: Multi-Agent Roundtable Tokens

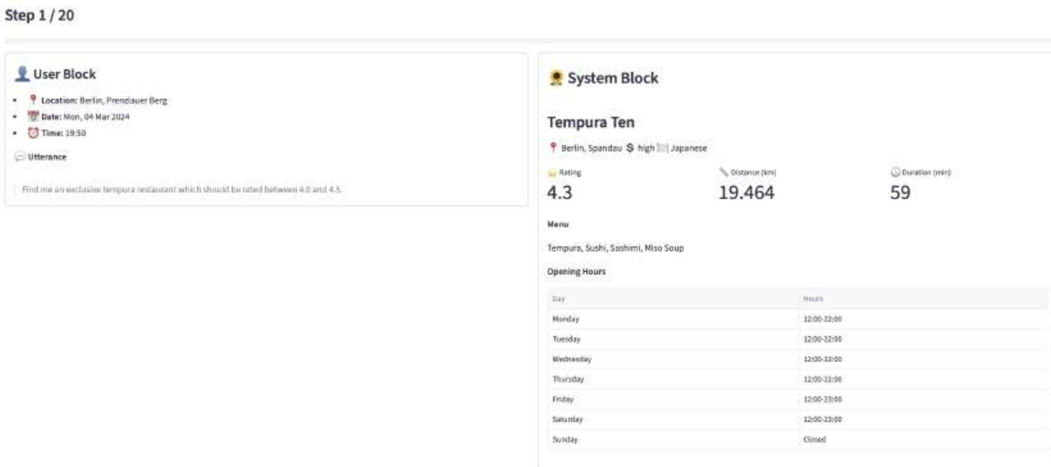


Figure B.29: Visualization of the user interface showing user and system blocks for the human agreement study (RQ0)

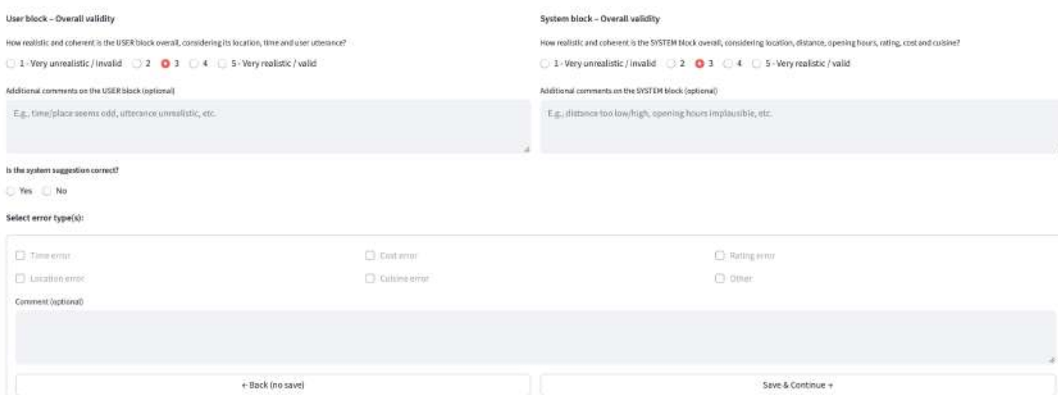


Figure B.30: Visualization of the user interface showing questions for the human agreement study (RQ0)