Combining Episodic Memory and LLMs for the Verbalization of Robot Experiences

Joana Plewnia and Tamim Asfour

***Abstract*— The ability to communicate past experiences is fundamental for intelligent and natural interaction. Humanoid robots continuously accumulate rich, multi-modal experiential data and must be able to articulate their episodic experiences in natural language to support effective human-robot communication. Existing approaches either rely on the generalization capabilities of large language models (LLMs), sometimes combined with episodic memory, or the precision of rule-based verbalization systems, each presenting limitations when used in isolation. In this work, we present a novel hybrid framework that integrates the adaptability of LLMs with the robustness of rule-based methods and the generalizable structure of memory-based approaches. Our system implements strategies to retrieve and transform memory representations of past perceptions and actions into natural language responses. This enables humanoid robots to respond to natural language queries about their experience. Experimental evaluation based on a set of distinct query types demonstrates that our approach successfully answers 89.4% of episodic memory questions with human-in-the-loop refinement, while reducing token consumption by 97% compared to pure LLM-based methods. Furthermore, we demonstrate the system's extensibility by leveraging LLMs, such as GPT-4.1, to expand the range of permissible queries through example-based interaction. The evaluation on our humanoid robot ARMAR-7 performing household tasks validates that our hybrid approach balances response quality with computational efficiency to address the crucial need for dependable yet flexible verbalization of robot experiences. Code and examples are available at robotexperienceverbalization.github.io/REV.github.io/**

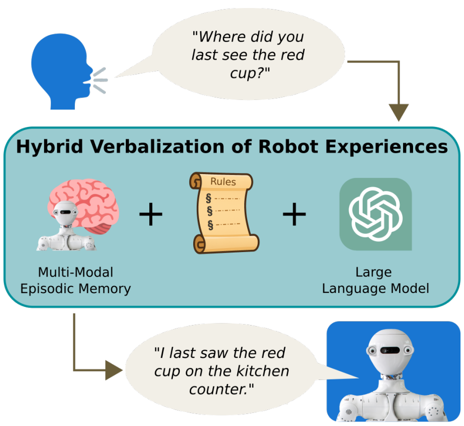

Fig. 1. We propose a hybrid verbalization method, combining episodic long-term memory, rule-based verbalization techniques, and large language models to enable a humanoid robot to verbalize its experience.

I. INTRODUCTION

Effective communication forms a cornerstone of successful interaction and collaboration, whether between humans or between humans and robots. This includes not merely conveying intentions and immediate observations, but also the ability to articulate and contextualize past experiences. For humans, verbal communication represents the most natural and intuitive means of communication, supporting collaboration, teaching, and social bonding. Therefore, to be effective collaborators, humanoid robots must be capable of verbalizing their past experiences in natural language, i. e., to transform their internal representations of past experiences to natural language narratives. This capability, termed *verbalization* by Rosenthal et al. [1], is defined as "the process of converting or narrating robot experiences via natural language". While the technical capacity to record and store experiential data has advanced considerably, the meaningful transformation of multimodal sensory data, action histories, and internal states into contextually appropriate natural language remains a significant challenge in cognitive robotics research.

In this work, we focus on enabling humanoid robots to articulate the contents of their episodic long-term memory (LTM) through natural language interaction. This task presents several challenges: it requires a structured and consistent framework for collecting, encoding, and storing past multimodal experiences, as well as a mechanism for transforming this structured internal representation into coherent and contextually appropriate natural language responses. Achieving both reliability in memory representation and variability in language generation presents a fundamental challenge that current approaches have not adequately resolved.

Contemporary solutions to this challenge generally follow two distinct paradigms: pure language model approaches that leverage the generative capabilities of large language models (LLMs) but often lack precision in factual recall, and rule-based systems that excel in accuracy but demonstrate limited adaptability to novel queries. In this work, we propose a hybrid system that combines the generalization capabilities of LLMs with the reliability of rule-based memory access mechanisms. Our approach enables humanoid robots to respond to a diverse range of questions about their experiences while dynamically extending their communicative repertoire through incremental learning from simple user-provided ex-

This work has been supported by the European Union's Horizon Europe Widening Program through the HERON project, the Carl Zeiss Foundation through the JuBot project, and the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG).

The authors are with the High Performance Humanoid Technologies Lab, Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Germany. E-mails: {joana.plewnia,asfour}@kit.edu

amples.

The main contributions of this work are twofold: (i) a hybrid framework for a memory- and LLM-based verbalization of robotic episodic knowledge, and (ii) a comprehensive architecture that demonstrates the applicability of this framework on a humanoid robot operating in a household environment.

The remainder of this paper is organized as follows: Section II briefly reviews prior work relevant to natural language verbalization of robot episodic memory. Section III introduces background information regarding the memory-centric cognitive architecture used in our approach to encode, store, and retrieve episodic knowledge, details our proposed hybrid verbalization approach, and describes the integration of the approach into our cognitive robot control architecture. Section IV presents experimental results on the humanoid robot ARMAR-7 and Section V concludes with a discussion of current limitations and future research directions.

II. RELATED WORK

A. Rule-based Verbalization

Early research on robot verbalization focused on translating a robot's internal plans or actions into human-understandable language using symbolic representations and manually crafted templates. Rosenthal et al. [1] introduced one of the first frameworks for robot self-narration by defining a *verbalization space* that allowed mobile robots to describe their navigation routes in different styles depending on user preferences.

Zhu et al. [2] extended these verbalization methods to a humanoid robot performing complex household manipulation tasks. They introduced a three-layer episodic memory model that captured experiences from low-level sensorimotor data to high-level task abstractions. Using stored logs and environment maps, the system generated natural language responses to queries (e. g., "What did you do in the kitchen?"). By combining symbolic planning with rule-based language generation, it narrated both navigation and manipulation actions during multi-step tasks like meal preparation.

Both early systems share a common structure, relying on predefined templates, automated log generation, and rule-based mechanisms to generate language, operating over symbolic data like waypoints or task logs. Overall, these early efforts demonstrated that robots can effectively narrate their experiences using symbolic internal representations and rule-based language generation. They also showed that explanation systems could be tailored to user needs. While limited in generalization beyond specific domains or robots, these systems laid the groundwork for later research by proving the feasibility and value of robot self-narration.

B. Memory-based Verbalization

More recently, research on robot verbalization shifted from rule-based methods to learning-driven approaches that integrate long-term memory systems and neural language generation techniques. Rothfuss et al. [3] introduced a Deep Episodic Memory system for encoding and storing rich, multi-modal data of a robot's actions and perceptions in a latent representation. Building on this, Bärmann et al. [4] developed an Episodic Memory Verbalization (EMV) model that used neural network architectures to generate natural language answers and summaries from stored episodes. This marked a shift from template-based approaches toward learned, flexible language generation capabilities. While these systems demonstrated improvements in verbalization quality and adaptability, they continue to face challenges related to memory scalability and substantial demands for training data.

C. LLM-based Verbalization

Recent advances in robot verbalization combine large pretrained language models (LLMs) with structured memory systems to create more fluent and accurate explanations of robot behavior. Two main trends have emerged: leveraging LLMs for interactive reasoning and language generation, and developing scalable memory systems to manage extensive long-term experiences.

DeChant et al. [5] fine-tuned a T5 model to generate summaries and answer questions about a robot's actions in simulation, showing that grounding LLMs in robot experience data improves both performance and generalization. Their work uses visual input and episodic metadata directly, without a systematic long-term memory storage. Anwar et al. [6] developed the ReMEmbR system, which enables real-world robots to respond to queries concerning activities spanning several hours using a retrieval-augmented memory system, producing both textual explanations and actionable responses to user inquiries. Similar to DeChant et al. [5], their approach only leverages visual information together with spatio-temporal metadata to construct a vector database as a memory mechanism, which is later used to supply an LLM with episodic information. Approaches like the one by Wang et al. [7] incorporate multiple information modalities beyond visual information. Their approach converts diverse input – including visual data, generated plans, and state information – into natural language representations generated by an LLM. A second LLM then generates a narration of those episodic summaries by aggregating the episodic summaries across different modalities. Recently, Bärmann et al. [8] introduced H-EMV, an advanced system that combines GPT-4 with a hierarchical memory tree to retrieve only the relevant segments from the long-term memory to supply the LLM during querying, maintaining both computational efficiency and response accuracy. The memory tree consists of several layers of summarization, with leaf nodes containing the episodic snapshots. Their work allows the use of a robot episodic long-term memory as an input for the hierarchical memory tree, on which an LLM directly acts to generate the answer to each question.

In these works, LLMs demonstrate exceptional language fluency and contextual understanding, while structured memory ensures factual grounding and temporal consistency. This synergistic combination enables robots to explain their experiences in natural language both during task execution and

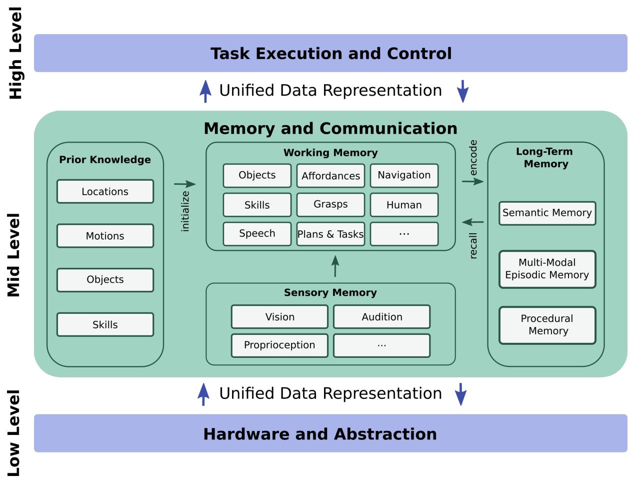

Fig. 2. Overview of the underlying memory architecture and its role in the robot control framework used as an example. Adapted from [9].

retrospectively. Despite these advances, current approaches face substantial challenges, including high computational costs, the need for large amounts of training data, difficulty in handling truly open-ended or unforeseen user queries, and the inherent limitations of LLMs to handle numerical operations.

D. Differences to previous Approaches

Our approach represents a novel hybrid approach that strategically combines elements of *memory-based*, *rule-based*, and *LLM-based* approaches. While previous *memory-based* approaches as in [3], [4] reply predominantly on learned representations of episodic knowledge and Transformer network architectures for natural language generation, we take a different approach, that directly accesses both symbolic and subsymbolic representations of experiential data and use template-based structures that yield more predictable outcomes while reducing training data requirements. Our approach also differentiates itself from existing *LLM-based* methods, which typically employ large language models for the complete answer generation to every query. In contrast, our approach invokes an LLM only to decide if a question matches a previously known pattern, to map parameters to template-based question structures, and to generate new question definitions when encountering a previously undefined query type. This reduces computational costs while enhancing the predictability and consistency of answers. Additionally, while most approaches that integrate long-term memory with *LLM-based* verbalization approaches require memory representations specifically tailored to their verbalization mechanisms, our approach offers greater architectural flexibility. Our system builds on top of an already existing memory system as part of a cognitive robot architecture, and thus does not require modification to the underlying episodic memory representation.

III. METHOD

A. Memory-based Cognitive Architecture

Our approach to verbalization falls into the category of *memory-based* approaches, leveraging a memory-centric cognitive architecture described in [9] with its memory system for encoding, storing, and episodic knowledge retrieval. An overview of this architecture can be seen in Figure 2. The memory system functions as an intermediate architectural layer, facilitating communication and abstraction between low-level hardware components and higher-level task planning and reasoning. Within this architecture, the episodic memory – a key component of the long-term memory – encodes experiential knowledge acquired during robot operation. This episodic memory captures discrete snapshots of the working memory at specific points in time, thereby enabling the construction of a temporal record of the robot's experiences. Access to this knowledge is organized hierarchically across multiple levels. At the top level, the memory architecture comprises distinct *memory servers*, each representing a semantically coherent collection of one or more sensor modalities. These memory servers contain multiple *segments*, which serve as modality-specific data containers. *Core segments* define standardized data types shared among all constituent elements, while *provider segments* are instantiated by client processes writing data into their associated core segment. Elements within provider segments are referred to as *entities*, which encapsulate representations of physical objects or abstract concepts capable of evolving over time. The dynamic state of an entity is captured through an *entity snapshot*, which may comprise a series of *entity instances*. At this most granular level, the data structure incorporates both the actual sensory data or symbolic information and associated metadata characterizing the entity at specific moments. The overall hierarchical organization of the memory system is illustrated in Figure 3.

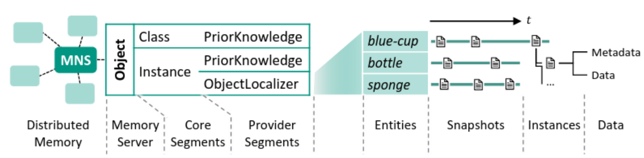

Fig. 3. Overview of the hierarchical structure comprising the memory architecture as described in [9]. An example from the Object *memory server* is shown, which consists of different objects and their associated data as *entity instances*.

B. Verbalization System

An overview of the verbalization system is shown in Figure 4. The system comprises two principal components: (1) a **Question Answering (QA) module**, responsible for generating answers to predefined questions, and (2) a **Question Generation (QG) module**, which creates new question definitions when the system encounters a question type it has not previously been configured to answer. Both modules are structured around three core strategies for answer generation, tailored to the nature of the question:

- The *Selection* strategy retrieves relevant information from the memory by identifying the relevant *memory server*, *segments*, and *entity* in which the answer to a given question is likely to reside.
- The *Search* strategy specifies the procedure for traversing the *entity snapshots* within the selected memory component.
- The *Evaluation* strategy determines whether a given *entity snapshot* contains the required information and outlines the process of extracting this information and converting it into symbolic, human-interpretable output.

In addition to the three strategies, the overall approach incorporates a *Match-Checker*, which determines whether an existing question definition can be used to answer a newly posed question. This matching relies on a *Question Template* – an abstracted version of the initial question, where specific phrases are substituted by placeholders. Correspondingly, an *Answer Template* is used to define how a response is constructed, incorporating placeholders for content derived from the robot's memory, provided via the *Evaluation* strategy.

Our verbalization system supports both manual and automatic definition of templates and strategies. Expert users may design these elements directly, or they may be generated using large language models like GPT-4.1 [10], given representative examples and knowledge of the memory structure and implementation. This allows the system to be initialized with a set of questions it can already answer, while also facilitating dynamic expansion of its capabilities at runtime through on-the-fly question definition. The system also supports a human-in-the-loop (HIL) approach, wherein a language model generates new question definitions that an expert user refines. An expert user is defined as someone knowledgeable about the memory system. The verbalization system is designed to be memory-system agnostic: each strategy can be flexibly defined to accommodate the specific characteristics of different episodic memory implementations. The only requirement for compatibility with our system is the presence of a memory system that stores snapshots, organized by entity and annotated with at least temporal metadata. The following sections detail the three core strategies, along with supporting components: the *Match-Checker*, and the *Question* and *Answer Templates*.

Question Templates and Match-Checker: A *Question Template* is a string pattern with optional placeholders, enabling generalization across semantically similar questions. Template complexity varies with question structure. At runtime, GPT-4o-mini [10] is used to match input questions and parameters to predefined templates. If no match is found, the *Question Generation* module prompts the model to create a generalized version by abstracting specific elements into placeholders and a completely new *Question Definition*, including Selection-, Search-, and Evaluation-Strategy implementations. To support this, the system provides contextual knowledge, including valid actions, objects, and entity names, allowing the model to distinguish variable from invariant components. When a

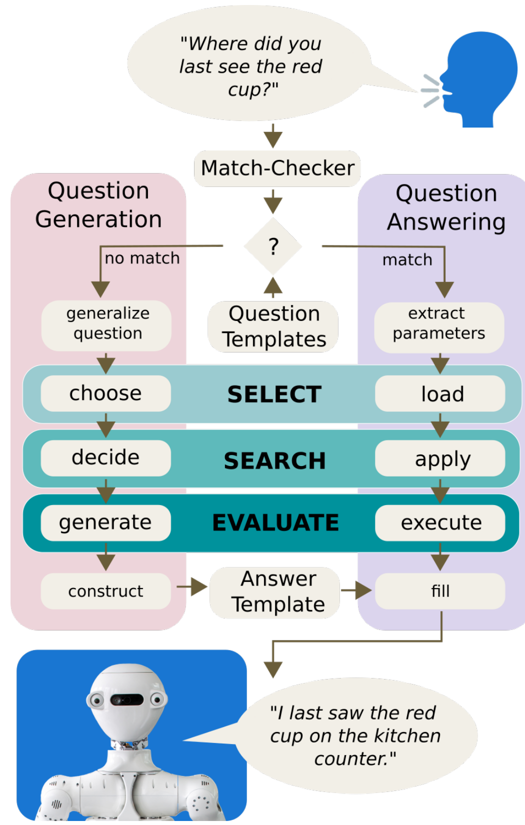

Fig. 4. Overview of our proposed method. The method comprises two main modules: the **Question Answering (QA)** module and the **Question Generation (QG)** module. The QA module utilizes an existing question definition to extract parameters from the input question, leveraging the *Question Template*, followed by the sequential execution of the *Select*, *Search*, and *Evaluate* strategies. The outcome of the evaluation step is used to populate an Answer Template, resulting in a natural language response. The QG module handles previously unseen questions by abstracting them into a *Question Template* for future matching. It then defines the corresponding *Select*, *Search*, and *Evaluate* strategies – optionally leveraging a large language model (LLM) – and generates a corresponding *Answer Template*.

match is identified, GPT-4o-mini binds the input question to the template by extracting relevant parameters. For instance, the question "Which objects did you see in the dishwasher the last time you opened it?" matches the template "Which objects did you see [0] the last time you [1]?", In this case, the placeholders would be instantiated with the values: [0] $\leftarrow$ "in the dishwasher", [1] $\leftarrow$ "open the dishwasher" The language model handles linguistic variability, such as anaphora, coreferences, and tense normalization, by leveraging contextual input, which improves template matching and parameter extraction.

Selection Strategy: The *Selection* strategy identifies relevant memory entities using a memory ID, which uniquely specifies an *entity* or a *memory segment* and thus a list of relevant entities. For the generation of new *Question Definitions*, we provide the language model with short explanations of the data stored in different segments of the memory, which, combined with semantically rich naming conventions of *entities*, allow the language model to infer relevance to a given question. The robotic software framework enables the retrieval of all snapshots associated with an entity via its memory ID, simplifying access to structured episodic memory. When a question requires information from multiple sources, the system can select multiple memory IDs sequentially. For instance, answering “Which objects did you see in the dishwasher the last time you opened it?” requires data from both a semantic scene and an action history entity snapshot, stored on separate memory servers. The selection strategy includes both memory IDs to ensure coordinated access for further processing.

Search-Strategy: To enable question-specific traversal of memory, we implemented multiple *Search* strategies for navigating the *entity snapshots* selected by the *Selection* strategy. These strategies adapt to the temporal semantics of a question and are currently limited to the predefined set given below. For example, questions with phrasing like “the last time you...” trigger a reverse chronological search, while references to specific time spans (e. g., “last week”) require filtering based on temporal metadata. The implemented search strategies include: *LAST* (reverse chronological search), *CURRENT* (latest snapshot only), *DURATION* (filtering within a defined interval), *EARLIEST* (chronological search), *BEFORE* (reverse chronological search from a defined point), and *AFTER* (chronological search from a defined point). For instance, to answer “What did you do today?”, the system applies the *DURATION* strategy to find all snapshots of high-level actions of the current day and retrieves their natural language names.

Evaluation-Strategy: The *Evaluation* strategy performs two main functions: determining whether an *entity snapshot* contains the information needed to answer a question, and extracting that information in human-readable form. Both are implemented as user-defined functions that operate on the snapshot’s content and metadata, allowing for domain-specific logic. When leveraging the automated generation of new *Question Definitions*, these functions have to be implemented by a language model using different pre-defined utility functions and free-form code. The first function evaluates a condition to assess the snapshot’s relevance. If satisfied, the second function extracts the necessary information, terminating the search. The extracted data is then used to populate the corresponding *Answer Template*. Both functions have access to all input parameters, including placeholder bindings from the associated *Question Template*. If n entities are part of this *Evaluation-Strategy*, the process is repeated n times, each time adding their generated output as an input to the next iteration. To answer a question such as ”Which objects did you see in the dishwasher the last time you opened it?”, the strategy first evaluates whether the currently examined snapshot in the action memory contains the action of opening the dishwasher. Upon identifying such a snapshot, the first iteration is concluded by providing the time at which the action took place. This timestamp is then used in a second iteration as an additional input to query the semantic scene memory. If the currently examined snapshot occurs sufficiently close in time, the strategy retrieves all objects whose semantic location aligns with ”in the dishwasher,” using sentence-transformers [11] for fuzzy semantic matching. The retrieved data from long-term memory is then used to generate the final natural language response.

Answer Template: The final step in both the *Question Answering* and *Question Generation* modules involves the *Answer Template*. Structurally similar to a *Question Template*, it is a string-based sentence containing placeholders for dynamic content. These placeholders may reference parameters extracted from the input question (e. g., [0], [1]), outputs from the *Evaluation* strategy (e. g., lists of object names), or a combination of both. The system supports list placeholders, which are automatically formatted as comma-separated sequences with “and” before the final item for natural language flow. For example, the template “The last time I [1] I saw [0]” may produce the sentence “The last time I opened the dishwasher I saw a red mug, a red plate, and a green bowl in the dishwasher,” assuming the corresponding observations were retrieved. This templating mechanism enables the system to generate fluent, contextually appropriate responses grounded in both the user’s query and the robot’s episodic memory.

IV. EXPERIMENTS AND EVALUATION

To validate our verbalization system and its implementation detailed in the section above, we ask episodic questions to our humanoid robot ARMAR-7 after it executed household tasks. This section details the evaluation process by explaining the experimental setup, the evaluation scenario, the episodic questions we asked, the evaluation metrics used, how we compare our proposed method to a purely LLM-based approach by Bärmann et al. [8], and the results of those experiments.

A. Experiment Setup and Evaluation Scenario

For evaluation, we consider the scenario where the humanoid robot ARMAR-7 performs typical household tasks such as object fetching, handovers, and environment inspection. Across five days over two months, we recorded 87 high-level task episodes with an average length of 9.89 minutes into their long-term memory, forming the dataset used in the evaluation. Each episode includes the robot’s internal state, visual data (RGB-stereo and depth), semantic and symbolic scene information generated online during robotic task execution, and transcribed speech input and output.

We use prompt parsing with GPT-4o-mini for frequent *Match-Checking* tasks and GPT-4.1 for the *Question Generation* module, leveraging its stronger reasoning capabilities. Details about the prompts can be found on the paper website[1]. For initial testing and demonstration, the language

[1] robotexperienceverbalization.github.io/REV.github.io/

model is guided by five manually defined question definitions, randomly selected from a set of six that represent different *Search* strategies, excluding the one corresponding to the current question. Expert users can define question behaviors via user-friendly templates that populate a predefined class structure encapsulating the *Select*, *Search*, and *Evaluate* strategies. The output of the *Question Answering* module is a natural language response, which can be verbalized using a robot-integrated text-to-speech (TTS) system.

B. Questions Types

We collected a set of questions about a robotic agent's episodic knowledge consisting of 17 natural language question types. They are composed of question types inspired by the Ego4D benchmark on Natural Language Queries for Episodic Memory [12], question types from Weston et al. [13] who identified question answering tasks to evaluate their AI-based QA system, and question types which users were curious about during demonstrations of our humanoid robots in our lab. From Ego4D, we excluded question types involving people – due to the absence of facial recognition –, questions about object properties – as those are not episodic –, and object state questions, as our system does not yet track such information. From this, we extracted important aspects of answering episodic knowledge questions: questions asking about episodic action information, episodic object localization, and basic object information, as well as questions that need counting, answering about before/after relations, and yes/no questions. From Weston et al. [13], we selected types that enriched our evaluation, including those requiring multi-source reasoning, multi-fact answers, and coreference resolution. While inspired by Ego4D and Weston et al., our final question set is tailored to a domestic humanoid robot context rather than egocentric human video data. The individual question count differs between categories, as some categories have a limited number of distinct options for questions and/or answers. This left us with a set of question types which can be seen in Table I.

C. Evaluation Metrics

We evaluate the approach and its implementation by using several key performance metrics across two distinct phases: (1) the generation of question definitions using an LLM-based mechanism, and (2) the subsequent answering of questions based on these definitions. For the first phase, we record the percentage of correctly generated question definitions (compared to the manually defined ground truth), the generation time, and the number of tokens consumed. For the second phase, we measure the accuracy of the generated answers, the accuracy of answers using the HIL approach, the time required to produce them, and the associated token usage.

This separation reflects the operational structure of the system, wherein the *Question Generation* module is a one-time, offline process, while the *Question Answering* module is used online during interactions with users. This distinction is critical, as the latter phase directly impacts real-time system performance and user experience.

D. Results

Table I presents the various types of questions evaluated, along with their corresponding performance metrics as defined in Section IV-C.

Question Generation Accuracy and HIL Approach: One can observe differences in the accuracy of the generated question definitions, ranging from question-types for which the generation always worked (e. g., L.1, A.5), to question types for which some generated definitions produced correct answers (e. g., L.4) and question types for which no definitions could be generated that yielded correct answers (e. g., all complex question types). In particular, the system encountered difficulties when generating definitions for questions that required the integration of information across multiple memory servers (e. g., C.1-C.3). This often led to challenges in determining the correct sequence of information extraction and usage in follow-up queries. Additionally, questions involving temporal relationships (see C.4) proved especially challenging for fully automatic generation. However, minor manual interventions by the author acting as an expert user significantly improved performance. For example, changes such as conversion of natural language expressions to internal identifiers (e. g., for objects, locations, or actions) contributed to an increase in the number of correctly answered questions using otherwise automatically generated definitions (e. g., A.3).

Computation Costs: The results indicate that while the generation of new question definitions incurs an average processing time of 37.84 seconds, the subsequent online phase of answering questions, based on these pre-defined templates, requires only 4.73 seconds on average. Notably, the time required to generate new question definitions varies significantly across question types. This variability is likely attributable to the complexity of the questions or their similarity to examples provided in the LLM prompts. The average response time for computing an answer was 4.73 seconds, though this metric was influenced by factors such as the size of the episodic memory, the temporal scope of the query (e. g., retrieving the most recent instance of an event), and the computational demands of the question definition itself. While queries about recent events (e. g., within hours or days) yield acceptable response times, those involving longer temporal spans (e. g., months) may incur significant delays, potentially degrading user experience.

Token Consumption: From a computational standpoint, the process of answering questions consumes an average of only 326 tokens. In comparison, the generation of new question definitions requires a substantially higher token count, with the answering phase representing merely 0.013% of the average token usage. This highlights that the bulk of complex reasoning and processing is carried out during the offline generation phase.

	Questions		Generation			Answering			
	Type	amount	Correctness	Time (sec)	Tokens	auto	HIL	Time (sec)	Tokens
L	**Robot & Object Location**								
1	Where were you at <point in time>?	6	100.0%	54.141	49289	100.0%	100.0%	2.512	270
2	Where did you last see <object>?	5	80.0%	60.285	49167	80%	90%	5.353	285
3	Which <affordance> objects do you have?	5	100.0%	61.510	49224	100.0%	100.0%	1.782	306
4	Which objects are at <location>?	6	66.6%	60.582	49282	60%	100.0%	2.067	262
5	When did you last see <object> at <location>?	2	100.0%	45.126	49504	100.0%	100.0%	2.465	315
6	Is <object> at <location>?	6	66.6%	35.008	88640	100.0%	100.0%	6.216	255
7	How many <object-type> are at <location>?	5	60.0%	61.487	38746	10.0%	100.0%	9.078	404
A	**Actions**								
1	What was the last thing you did?	5	60.0%	16.104	26620	100.0%	100.0%	3.846	306
2	When did you last do <action>?	3	66.6%	11.355	21925	20.0%	50.0%	15.104	289
3	What did you do today?	6	33.3%	12.921	5224	100.0%	100.0%	2.688	333
4	What did you do during <timespan>?	3	66.6%	18.062	30839	100.0%	100.0%	2.691	327
5	You grasped <object> today. When did you grasp it?	2	100.0%	10.537	16712	100.0%	100.0%	4.420	288
6	How often did you do <action> during <timespan>?	3	66.6%	17.483	21417	33.3%	100.0%	6.170	354
C	**Complex Questions**								
1	What did you <action> in <location>?	6	0%	-	-	0%	100.0%	3.202	381
2	Where did you <action> the <object>?	4	0%	-	-	0%	100.0%	4.505	349
3	When you last heard <sentence>, where were you?	3	0%	-	-	0%	100.0%	2.307	376
4	What did you do after/before <action>?	6	0%	-	-	0%	60.0%	5.132	368

TABLE I

RESULTS OF VERBALIZATION EXPERIMENTS

Accuracy of Answers: The table also includes accuracy metrics for the answered questions. When relying solely on automatically generated question definitions, the system achieved a mean success rate of 53.12%. However, incorporating human-in-the-loop adjustments and human generation of question definitions where automatic generation processes failed boosted the average success rate to 89.41%. Persistent challenges included questions involving action parameters (e. g., specifying the object involved in a ”grasp” action), as these introduced considerable variability within ostensibly similar question definitions. Similarly, resolving natural language references to unique object identifiers is challenging in uniform datasets with similar entities, underscoring the need for a coherent and semantically structured memory for accurate querying.

E. Comparison to pure LLM-based Approach

In addition to evaluating our own method, we conducted a comparative assessment using an alternative approach following the methodology proposed by Bärmann et al. [8], applying it to a subset of the same set of questions. The use of a subset was necessitated by the substantial token and computational resource requirements associated with their approach, as observed during preliminary experimentation. We selected 12 representative question types targeting past actions, object locations, and auditory inputs.

Accuracy of Answers: The pure LLM-based approach on average yielded 5 fully correct answers and 2 additional answers that were partially correct – i. e., they contained the correct response alongside extraneous inaccurate information. In comparison, our system correctly generated 4 question definitions on the first attempt, and an additional 5 definitions that required only minor modifications to yield accurate answers. The alternative approach demonstrated particular strength in cases where the queried information constituted a central component of an action, for instance, identifying the object involved in a grasping action or articulating the description of an executed behavior. However, it encountered difficulties when addressing queries that required counting events or retrieving information considered peripheral to the core action (e. g., incidental auditory input perceived during task execution). Our approach showed clear advantages in these latter scenarios. It effectively handled counting tasks (e. g., determining the number of times a specific action was performed within a defined time window), accurately responded to questions regarding secondary sensory inputs (such as non-instructional auditory data), and demonstrated high consistency and reliability in repeated question types. Once a question definition is correctly formulated and covers the common cases, our method consistently provides accurate responses with a high degree of certainty. Notably, our approach avoids producing partially correct answers. Due to the formalized structure governing information extraction, the system either retrieves the necessary data correctly or fails without generating superfluous content. Nonetheless, a limitation of our system lies in the capability of GPT-4.1 to autonomously generate accurate question definitions for more complex queries. Additionally, the rigid formalism used in our method constrains its flexibility when answers may need to be synthesized from unexpected sources. Both systems occasionally misinterpreted question types, often conflating ”What,” ”When,” and ”Where” formulations, leading to semantic mismatches in the responses.

Computational Costs: The pure LLM-based approach consumed an average of 48200 tokens per question and 22156 tokens daily for summarizing prior actions. While token usage can be reduced through caching, this is only applicable if no new actions have modified the underlying data. In contrast, our method required only 418 tokens per question and 25071 tokens to generate a new question definition. Similar differences were observed in computation time: the LLM-based method averaged 14.83 s per question (max: 80.88 s) and 2.1 s per action taken by the robot to update the history tree, whereas our system averaged 4.76 s per question (max: 15.48 s), incurred no cost during memory updates, and required 39.51 s on average to define new question types. The pure LLM-based approach shows high latency and token costs, particularly for questions concerning experiences not previously marked as significant for summarization, requiring full history traversal. In contrast, our method's cost scales primarily with memory size, temporal distance of the queried event, and question relevance.

V. CONCLUSION

We presented a hybrid framework combining rule-, memory-, and LLM-based verbalization of episodic knowledge, to enable reliable and efficient question answering grounded in structured memory representations. The system demonstrates the ability to handle object- and place-related questions from the Ego4D benchmark, as well as more complex queries that require integrating multiple supporting facts. In addition, it supports counting tasks and can verbalize peripheral information, such as auditory input, that was not explicitly utilized elsewhere in the cognitive pipeline. Compared to purely LLM-based approaches, our method is significantly more computationally efficient and delivers more consistent results by operating over semantically structured episodic memory and leveraging rule-based answering templates. Nonetheless, this comes with trade-offs. While LLMs naturally produce descriptive language for actions, objects, and scenes, our approach benefits heavily from manual specification of spoken labels for each object and action ID. Moreover, the generation of new question definitions is limited by the reasoning capabilities of GPT-4.1, and our current template-matching mechanism still struggles to robustly disambiguate between different question types such as "what," "when," and "where."

While the proposed approach demonstrates the viability of a general framework for verbalizing robotic episodic knowledge, several avenues for further research remain. In contrast to prior work (e. g., [1], [2]), our system does not yet incorporate personalization or adaptive verbalization strategies, which could enhance user engagement and perceived system intelligence. Additionally, the evaluation has so far been limited to a single robotic platform, which restricts generalizability. Future work should also include broader empirical validation using standardized datasets such as Ego4D. However, these datasets are outside of the robotic domain, presenting further challenges for evaluation generalizability. To expand the framework's capabilities, future iterations may incorporate mechanisms for querying semantic memory alongside episodic content, thereby broadening the range of supported interactions. While our current implementation avoids visual question answering (VQA) in favor of runtime image-based processing, the framework remains fully compatible with VQA techniques, offering a promising extension to enhance visual reasoning and perception-based dialogue.

REFERENCES

[1] S. Rosenthal, S. P. Selvaraj, and M. M. Veloso, "Verbalization: Narration of autonomous robot experience." in *IJCAI*, vol. 16, 2016, pp. 862–868.

[2] Q. Zhu, V. Perera, M. Wächter, T. Asfour, and M. Veloso, "Autonomous narration of humanoid robot kitchen task experience," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2017, pp. 390–397.

[3] J. Rothfuss, F. Ferreira, E. E. Aksoy, Y. Zhou, and T. Asfour, "Deep episodic memory: Encoding, recalling, and predicting episodic experiences for robot action execution," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 4007–4014, 2018.

[4] L. Bärmann, F. Peller-Konrad, S. Constantin, T. Asfour, and A. Waibel, "Deep episodic memory for verbalization of robot experience," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 3, pp. 5808–5815, 2021.

[5] C. DeChant, I. Akinola, and D. Bauer, "Learning to summarize and answer questions about a virtual robot's past actions," *Autonomous robots*, vol. 47, no. 8, pp. 1103–1118, 2023.

[6] A. Anwar, J. Welsh, J. Biswas, S. Pouya, and Y. Chang, "Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation," *arXiv preprint arXiv:2409.13682*, 2024.

[7] Z. Wang, B. Liang, V. Dhat, Z. Brumbaugh, N. Walker, R. Krishna, and M. Cakmak, "I can tell what i am doing: Toward real-world natural language grounding of robot experiences," *arXiv preprint arXiv:2411.12960*, 2024.

[8] L. Bärmann, C. DeChant, J. Plewnia, F. Peller-Konrad, D. Bauer, T. Asfour, and A. Waibel, "Episodic memory verbalization using hierarchical representations of life-long robot experience," *arXiv preprint arXiv:2409.17702*, 2024.

[9] F. Peller-Konrad, R. Kartmann, C. R. G. Dreher, A. Meixner, F. Reister, M. Grotz, and T. Asfour, "A memory system of a robot cognitive architecture and its implementation in armarx," *Robotics and Autonomous Systems*, vol. 164, pp. 1–20, 2023.

[10] OpenAI, "GPT-4 Technical Report," *arXiv:2303.08774*, 2023.

[11] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[12] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, M. Martin, T. Nagarajan, I. Radosavovic, S. K. Ramakrishnan, F. Ryan, J. Sharma, M. Wray, M. Xu, E. Z. Xu, C. Zhao, S. Bansal, D. Batra, V. Cartillier, S. Crane, T. Do, M. Doulaty, A. Erapalli, C. Feichtenhofer, A. Fragomeni, Q. Fu, C. Fuegen, A. K. Gebreselasie, C. González, J. M. Hillis, X. Huang, Y. Huang, W. Jia, W. Khoo, J. Kolár, S. Kottur, A. Kumar, F. Landini, C. Li, Y. Li, Z. Li, K. Mangalam, R. Modhugu, J. Munro, T. Murrell, T. Nishiyasu, W. Price, P. R. Puentes, M. Ramazanova, L. Sari, K. K. Somasundaram, A. Southerland, Y. Sugano, R. Tao, M. Vo, Y. Wang, X. Wu, T. Yagi, Y. Zhu, P. Arbeláez, D. J. Crandall, D. Damen, G. M. Farinella, B. Ghanem, V. K. Ithapu, C. V. Jawahar, H. Joo, K. Kitani, H. Li, R. A. Newcombe, A. Oliva, H. S. Park, J. M. Rehg, Y. Sato, J. Shi, M. Z. Shou, A. Torralba, L. Torresani, M. Yan, and J. Malik, "Ego4d: Around the world in 3,000 hours of egocentric video," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18 973–18 990, 2021.

[13] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv: Artificial Intelligence*, 2015.