Adaptive Task Scheduler for Social and Intelligent Assistive Robots in Home Environment

Priyank Avijeet
Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
pavijeet@uwaterloo.ca

Vinayak Gajendra Panchal
Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
vpanchal@uwaterloo.ca

***Abstract*—In the evolving landscape of home automation, the integration of Social and Intelligent Assistive Robots is becoming increasingly prevalent. Anticipating a future where these robots perform a variety of tasks to assist in daily human life, our project focuses on the crucial aspect of task scheduling. The unique challenge lies in the non-sequential nature of task execution, mirroring the human approach to managing daily activities based on priority, context, and urgency. We have introduced an advanced system that orders tasks based on a priority score, calculated by analyzing emotional context, health sensitivity, and temporal aspects. To address this, we have developed an innovative interface that leverages Natural Language Processing (NLP) and Transformer models to extract and interpret task-related information from user inputs. The core of our system is a self-adaptive software architecture, employing the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) loop to adjust the prioritization of tasks dynamically. Additionally, by analyzing data over time, our system can estimate different objects and environments the robot should focus on throughout the day, indicating adaptations to the user's behaviour. We also adjust the emotion preference weights based on user input trends, adding a layer of emotional intelligence to task management. Our project enhances Human-Robot Interaction by enabling robots to intelligently and empathetically manage tasks, demonstrated through a Graphical User Interface that visualizes the robot's task stack. This advancement not only improves task execution but also brings a more empathetic and responsive dimension to robotic assistance in everyday life.**

***Index Terms*—Social and Intelligent Assistive Robots, Self-Adaptive Software Systems, Natural Language Processing, Transformers, Human-Robot Interaction**

I. Introduction

In the near future, we can envision a world where social robots seamlessly integrate into our lives, taking over the mundane daily chores that often consume our time and energy. This vision forms the cornerstone of our project, which aims to bring about a significant improvement in Human-Robot Interaction (HRI) by making the task scheduling process self-adaptive. Our service robots are proficient in carrying out tasks requested by users, focusing on one pre-planned task at a time [1]. Unlike traditional task scheduling methods, our approach is inspired by the intuitive, flexible way humans plan their days. We constantly adjust our schedules based on a myriad of factors, including urgency, personal preference, and changing circumstances. Capturing this dynamic and adaptable aspect of human behaviour in the functionality of social robots is the primary goal of our endeavour.

At the heart of our project [1] is the understanding of the basis on which humans prioritize and reorder tasks. After considerable reflection and research, we identified three pivotal elements that guide human decision-making: emotions, health sensitivity, and temporal considerations. These elements are crucial in shaping our approach to task scheduling, and we have sought to emulate them in the behaviour of social robots. Our system employs an innovative method where instructions to robots are given using both text and emojis, allowing for the capture of emotional context. A dynamic task list is then displayed on an intuitive GUI, where users can influence task prioritization by adjusting the weights for happy, sad, and neutral emotions.

Our design and theme for the solution of ordering the task list based on priorities is inspired by preemptive and priority-based task scheduling in Operating systems [2]. In preemptive priority task scheduling, tasks are executed based on their priority levels, ensuring that higher-priority tasks are given precedence over lower-priority ones. This approach aligns with the way humans naturally manage their daily activities, where urgent and important tasks take precedence. By incorporating this concept into our system, we prioritize tasks not only based on their inherent urgency but also by considering emotional context, health sensitivity, and temporal aspects. This prioritization strategy mirrors the cognitive processes of humans, enabling our system to make more informed and context-aware decisions.

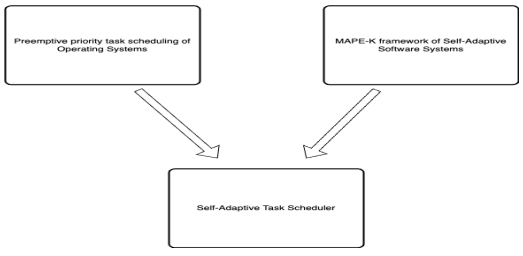


Fig. 1. Our approach for designing Self-Adaptive Task Scheduler.

[1]https://github.com/prikavj/Adaptive-Task-Scheduler-for-Social-and-Intelligent-Assistive-Robots-in-Home-Environment

The justification for this design choice lies in its ability to enhance the efficiency of task execution in a dynamic and adaptive environment. Operating systems have successfully employed preemptive scheduling to optimize resource allocation and responsiveness for years. Adapting these principles to our project enables our robots to intelligently manage tasks, respond to user needs promptly, and provide a more empathetic and user-centric approach to task execution. By drawing from the field of operating systems, we leverage proven strategies to create a home automation system that not only automates tasks but also understands and adapts to the complexities of human life, ultimately improving the user experience.

The core of our project involves an advanced algorithm, akin to a utility function in a MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) loop [3], which dynamically adapts the robot's task list by calculating a priority score for each task. This process considers the emotional weightings assigned by the user. In a novel approach, the system also adapts these emotion weights based on the dominant emotion in recent user inputs, ensuring tasks deemed important by the user are given precedence.

Beyond mere task management, our project extends to context and temporal awareness. By analyzing data patterns over time, we infer which objects the robot should be attentive to at various times of the day. This could also extend to estimating the robot's optimal locations throughout the day, ensuring it is always well-positioned to respond to the user's urgent needs promptly.

Another aspect of our project is the creation of a mood model of the user based on sentiment analysis of their inputs. This feature enables the robot to not only perform tasks but also to interact in ways that could help improve or stabilize the user's emotional state. This approach is inspired by the empathetic interactions typical among family members, aiming to position the robot as a sensitive and responsive member of the household.

Our project is not just about creating robots that can perform tasks; it's about developing a system where robots understand, adapt, and respond to the nuances of human emotions and needs. Current times mark the beginning of a new era in HRI, where robots are more than mere machines—they become empathetic, adaptive, and integral components of enhancing the quality of our daily lives. The potential impact of this technology is vast, promising a future where the synergy between humans and robots reaches new, unprecedented levels.

II. Methodology

A. Overview

Our project aimed to demonstrate the effectiveness of a self-adaptive system architecture. The platform of this demonstration was the development of a web application capable of processing text inputs with emojis and organizing them into a task list. This section outlines the methodology used in the design and implementation of this system.

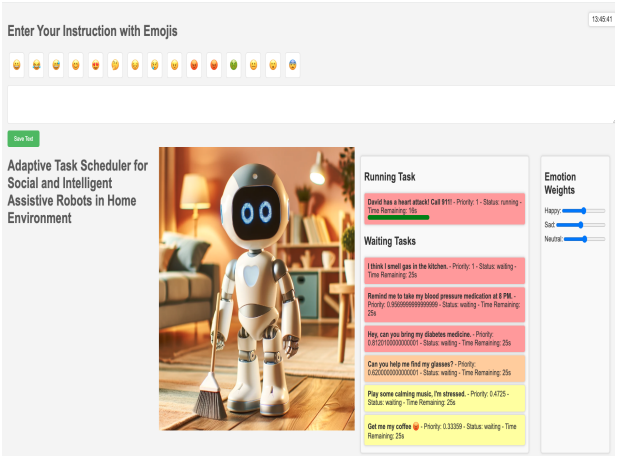


Fig. 2. User Interface for Adaptive Task Scheduler for Social and Intelligent Assistive Robots in Home Environment.

B. Web Application Development

1) Frontend Design: The web application's frontend was crafted using HTML, CSS, and JavaScript. These technologies were chosen for their robustness and wide usage in web development, ensuring an interactive and user-friendly GUI. The primary functions of the frontend included:

- Capturing user inputs, including text and emojis.
- Displaying the organized task list in a coherent and accessible format with progress bars mimicking the task completion state.
- Ensuring responsive design for usability by having sliders to adjust emotion weights that impact the task order.

2) Backend Implementation: Python, with its Flask framework, was the backbone of our backend development. Python Flask is known for its simplicity and efficiency in handling web server tasks. The backend was responsible for:

- Processing and storing user inputs from the frontend.
- Our system efficiently handles the logic behind organizing and updating task lists. Additionally, it keeps track of the user's recent emotions, which are visually represented through the robot's facial expressions on the user interface. Furthermore, the user's emotional state dynamically influences the weight sliders for emotions, allowing for a more personalized experience.
- Handling data retrieval and storage operations.

C. Data Management

1) State Data Storage: The state data, including the current status of tasks and user interactions, was stored in JSON format. JSON was selected due to its lightweight nature and ease of integration with Python and JavaScript. The JSON data is only for the current interaction session to display the task list in the UI and not for the past user interaction sessions that could be used for inference and analysis.

2) User Data Analysis: For analyzing user interactions and system performance, we stored user data in CSV files for long-term use. This format was chosen for its compatibility

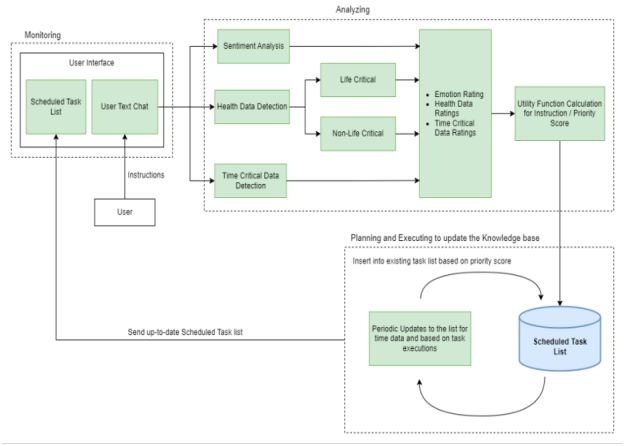


Fig. 3. Block Diagram of different components in our design.

with data analysis tools, enabling comprehensive analysis and insights derivation on system usage and user behaviour.

D. Project Scope and Context

This project serves as a preliminary step in exploring the architecture of self-adaptive systems in this context. The web application forms the basis for testing and refining self-adaptive mechanisms that will be integrated into the system. Our approach provides a high-level view of the implementation and field settings.

E. Self-Adaptation Architecture and MAPE-K Framework for Task List

In this section, we present the methodology behind our self-adaptation architecture, which is fundamental to our system's ability to dynamically respond to changing user needs and environmental conditions in real-time. Our methodology is inspired by and built upon the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) loop framework, and it plays a pivotal role in various facets of our system, especially in task scheduling and prioritization. Figure 3 shows how different components interact with each other in our designed system.

1) Uncertainties in System Adaptation: To ensure effective adaptation, our system addresses several uncertainties:

- Emotional Demands: We consider the emotional state of the user, including emotions such as happiness, sadness, and neutrality, to influence task prioritization.
- Health Emergencies: Prioritization takes into account life-critical medical emergencies and non-emergency medical needs.
- Time Sensitivity: The system considers the requested delivery time for tasks and adjusts their priority as the scheduled time approaches.

2) Adaptive Goals:

- Emotion-Based Adaptation: Our system incorporates emotion-based adaptation goals. Users can assign weights to different emotional states (e.g., happy, sad, neutral), and these weights influence task prioritization. Furthermore, we dynamically adapt emotion weights over time to give higher priority to the most recent emotional states, effectively addressing the user's current emotional needs.
- Health-Related Adaptation: We classify tasks into three categories: "life-critical medical emergency," "casual medical need," and "general activity." This classification aids in prioritizing tasks that directly impact the user's health and well-being.
- Time-Sensitive Adaptation: Tasks are scheduled and prioritized based on their requested delivery time. As the scheduled time approaches, the system dynamically adjusts task priorities to ensure timely completion.

3) Implementation: Adaptive Task List: A critical feature demonstrating our system's adaptability is the Adaptive Task List. It adheres to the MAPE-K paradigm:

1) **Monitoring (M):** User instructions are collected through a GUI with text input and emojis, stored in JSON files for analysis.
2) **Analysis (A):** The Analyzer component processes tasks every second using Natural Language Processing (NLP) and Transformers. It focuses on sentiment extraction and health data classification.
3) **Planning (P) and Execution (E):** The task list is continuously updated, reordered, and executed based on the components discussed above. The system's knowledge base remains up-to-date to align with the defined Adaptation Goals.

4) Analyzing User Instructions: The Analyzer component performs the following analyses for each task:

1) **Sentiment Analysis:** We extracted the "emotion_score" data to analyze emotions in user-provided text. Recognizing that many emotion recognition models struggled with emojis, we implemented a solution to convert emojis into their corresponding text descriptions and combined them with the original user text. The most effective model, "emotion-english-distilroberta-base" by J Hartmann [4], excels at identifying emotions such as joy, surprise, sadness, fear, anger, disgust, and neutral. It assigns scores to each emotion and then categorizes them into "happy," "sad," and "neutral" through a normalization process of clubbing positive emotions as happy and negative emotions as sad for simplicity. This approach harnesses the efficiency of DistilRoBERTa [5] models while providing nuanced emotion detection capabilities, making the "emotion_score" a powerful parameter for sentiment analysis.
2) **Health Data Classification:** We developed a module named health_text_classifier designed to categorize user-provided text into three different categories: "life-critical medical emergency", "casual medical need", and "general activity". We initially experimented with various machine learning classification models to distinguish between "life-critical medical emergencies" and "casual medical emergencies." However, we shifted our

focus to a superior approach, utilizing the "bart-large-mnli" model [6] [7] from Facebook, which is fine-tuned for zero-shot classification. This model combines the strengths of bidirectional and auto-regressive architectures, enhancing context understanding and prediction coherence. This combination of BART's [7] powerful architecture and MNLI's [8] diverse training makes our module highly effective in classifying the severity and nature of health-related scenarios instances, like life-threatening emergencies and casual medical situations based on user textual tasks. It provides probability scores for each class, with higher scores indicating greater confidence in the classification.

3) **Priority Score Calculation:** We compute a priority score for each task, taking into account emotional and health sensitivity, using the formula:

$$\text{Priority Score} = 0.65 \times \text{Total Emotion Score} + 0.35 \times \text{Health Sensitivity Score} \quad (1)$$

We, as stakeholders, determined the preference scores through experimentation with our tool.

4) **Handling Special Cases:** Life-threatening emergencies (life-critical medical emergencies score > 0.30) receive the highest priority, while temporal aspects are managed by scheduling tasks and adjusting their priority 10 seconds before the requested time.

Our comprehensive approach to task scheduling and prioritization within the self-adaptation architecture exemplifies the system's adaptability. It represents a significant advancement over traditional task execution models, adding logic and efficiency to the complex domain of home automation task management. Integration with the MAPE-K loop ensures continuous monitoring and adjustment of tasks, aligning the system's operations with the dynamic needs and preferences of the user.

F. Three-Layered Self-Adaptive Software Architecture

Figure 4 illustrates a three-layered self-adaptive software architecture [9] for the Adaptive Task Scheduler in Intelligent Assistive Robots, focusing on dynamic task scheduling. This system is designed to handle the complexities of daily human activities, which require non-sequential task execution based on varying priorities, contexts, and urgencies. The architecture comprises three main layers:

1) Component Control Layer: Positioned at the foundational level, this layer engages directly with the environment. It serves as the initial point where user tasks are collected and sorted. Within this layer, there are three primary elements:

a) Text box with emojis: Acts as a sensor for all user-submitted tasks via the user interface, awaiting further scrutiny. Tasks are then elevated to a higher layer for detailed analysis and required modifications.

b) Scheduled Task List: Represents the rearranged or prioritized tasks subsequent to the application of adaptive strategies.

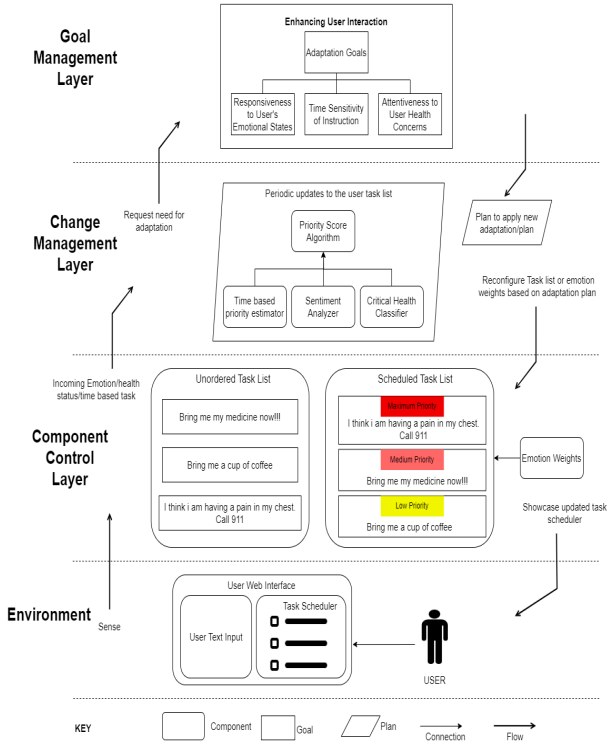


Fig. 4. Three-layer Architecture model.

c) Emotion Weights: Assigns varying levels of importance to each emotion, indicative of their significance. These weights are adjusted according to the user and the adaptations needed.

2) Change Management Layer: Situated as the intermediary stratum, this layer is responsible for modifying the Component Control Layer in response to perceived changes. It encompasses:

a) Sentiment Analyzer: Assesses the emotional content of user instructions.

b) Critical Health Classifier: Gauges the urgency of health-related tasks.

c) Time-based Priority Estimator: Orders tasks by their submission time or urgency.

The outputs from each of these analytical tools are then consolidated to determine the tasks' utility values through the "Priority Score Algorithm".

3) Goal Management Layer: At the top hierarchy, this layer sets "adaptation goals" to enhance user interaction. It ensures the system is responsive to the user's emotional states, sensitive to the timing of instructions, and attentive to health concerns. Strategies for adaptation are crafted following an analysis of the circumstances, which are then relayed to the Change Management Layer to be put into action.

a) Flow of Activities: In the Adaptive Task Scheduler for intelligent assistive robots, user tasks are initially inputted into the system via a web interface and collected through the Component Control Layer. These tasks are then assessed for emotional, health urgency, and time sensitivity by the Change

Management Layer, which employs a Priority Score Algorithm to assign a priority score to each task. The Goal Management Layer reviews these scores against the system's high-level adaptation goals, formulates a plan for task prioritization, and communicates it back to the Change Management Layer for execution. Tasks are dynamically scheduled and updated in the Scheduled Task List and Emotion Weights are adjusted according to their assigned priority, with the system continuously monitoring for changes in user input and state to adjust the scheduling as needed, ensuring that the robot adjusts empathetically and efficiently to the user's needs.

The architecture enhances Human-Robot Interaction (HRI) by enabling robots to handle tasks while considering the user's emotional, temporal, and health needs. It aims for robots to complete tasks empathetically, aligning with the user's priorities and preferences. A Graphical User Interface displays the robot's tasks and their dynamic prioritization based on the system's analysis.

G. Self-Adaptation for Emotion Weights

As we were already collecting the emotion scores and storing them in a CSV file, we implemented a logic to increase the weight of emotions that seem to be predominant in the instructions in the near future. The idea behind this is that if a person consistently expresses a specific emotion, such as anger, across multiple instructions, the system should adapt by assigning higher priority to tasks associated with negative emotions, thereby addressing the user's needs more promptly.

To achieve this, we devised an update formula:

$$\begin{aligned}&\text{if emotion == current\_emotion:}\\&\quad \text{emotion\_weights[emotion]} =\\&\qquad current\_weight + (100 - current\_weight) * 0.3\\&\text{else:}\\&\quad \text{emotion\_weights[emotion]} = current\_weight * 0.7\end{aligned} \tag{2}$$

This logic is run at intervals and the current_emotion is the average emotion score during that interval. Here the update value is 30% of the distance from the 0 or 100 boundary. The formula was designed such that with each update as we get closer to 0 or 100 the magnitude of the update gets smaller, keeping the updated value bounded. This adaptive mechanism enables the system to dynamically adjust emotion weights based on the user's emotional state, ultimately leading to a more personalized and responsive user experience.

III. Obtained Results

In developing our self-adaptive task scheduler, we employed an incremental model of software development [10].

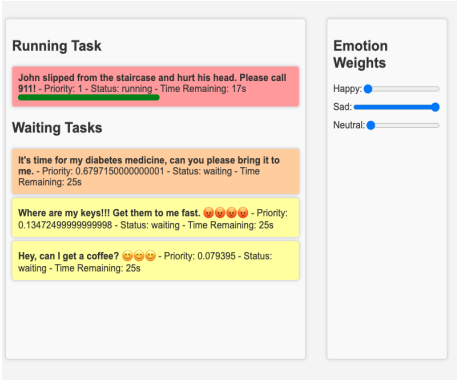
A. Incremental Model of Software Development

This approach allowed us to build and refine the platform in distinct, manageable stages, each adding functionality and complexity. Below, we detail the key phases of this development process:

1) **Initial Web Application Development:** The first stage involved creating a basic web application. This platform served two primary functions: it allowed users to input tasks, and it stored these tasks on a backend server. The user interface was designed to enable users to visualize their ordered task list, enhancing interaction and usability.
2) **Progress Bar Implementation and Real-Time Updates:** To simulate task execution, we integrated progress bars that represented the ongoing tasks. These bars were dynamically updated every second, fetching the latest data from the server to provide real-time feedback to the user.
3) **Emotion-Based Task Ordering:** The subsequent phase introduced an adaptive layer where tasks were prioritized based on emotional context and their associated preference weights. This feature marked the beginning of our system's ability to adapt to the user's emotional state.
4) **Incorporating Time Sensitivity:** We then enhanced the system to consider time-sensitive aspects of tasks. This iteration allowed the scheduler to prioritize tasks not only based on emotional context but also on their urgency and temporal importance.
5) **Health Sensitivity and Emergency Context Integration:** The next step involved classifying tasks based on health sensitivity and emergency contexts. This integration allowed the system to further refine task prioritization, considering critical health-related tasks as a priority.
6) **Adaptive Emotion Weight Sliders:** We introduced an adaptive element to the emotion weight sliders in the user interface. The system was designed to increase the weights of predominant emotions in the recent context, based on sentiment analysis of user instructions in the past interval.
7) **Data Inference and Visualization for Daily Routine Optimization:** In our pursuit of efficiency, we began to infer patterns from text instructions, which included emojis. This analysis aimed to identify objects and environments that users frequently interacted with. For instance, identifying a pattern of searching for car keys in the morning accompanied by stress-related emojis. We visualized this data to infer focusing on specific objects at certain times of the day. This insight could also inform future enhancements, such as positioning the robot in strategic locations at specific times to improve service efficiency, like being near the hallway when the user is preparing to leave for work.

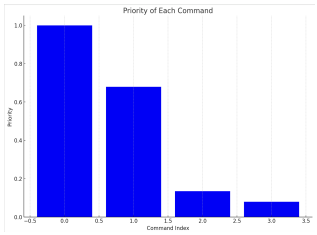
This incremental development approach not only facilitated a structured and systematic build-up of the system's capabilities but also allowed for continuous testing and refinement at each stage. The end result is a sophisticated, user-responsive task-scheduling system that intelligently adapts to both the emotional and practical needs of the users.

(a)

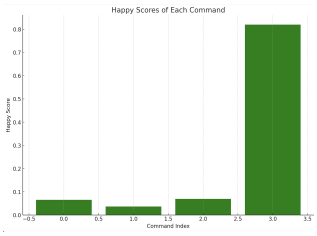


(b)

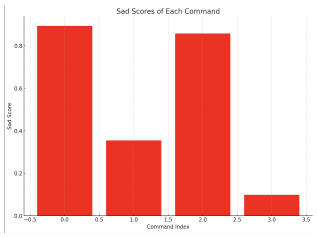
Fig. 5. (a) Unordered task list (b) Ordered task list using our self-adaptive task scheduler.



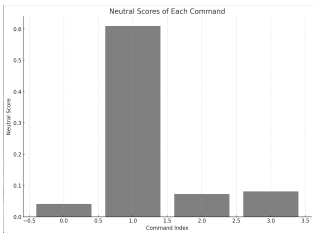
(a)



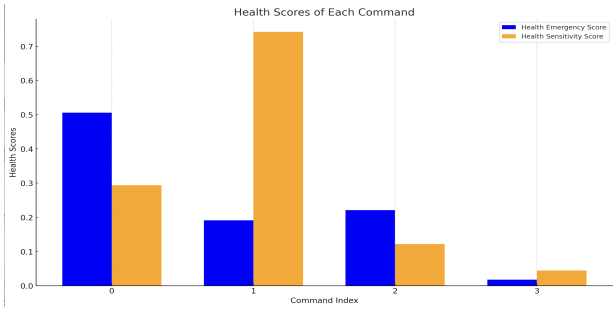
(b)



(c)



(d)



(e)

Fig. 6. Graphs for the various ratings given by our system (a) Priority Score, (b) Happy emotion score, (c) Sad emotion score, (d) Neutral emotion score, and (e) Health scores for each command.

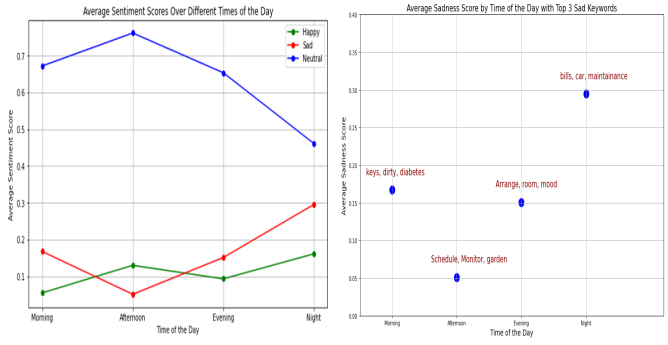
B. Results and Discussion

To gain a clearer understanding of our results, we will adopt an example-based approach for demonstration purposes. This will involve taking a set of instructions, feeding them into our system, and then analyzing the outcomes. A list of sample commands and their respective indices from the ordered task list are as follows:

- **Index 0:** "John slipped from the staircase and hurt his head. Please call 911!"
- **Index 1:** "It's time for my diabetes medicine, can you please bring it to me."
- **Index 2:** "Where are my keys!!! Get them to me fast."
- **Index 3:** "Hey, can I get a coffee?"

The emojis incorporated into the commands are illustrated in Figure 5. This figure presents two distinct scenarios. In Figure 5(a), we observe an unordered list representing a system that operates without self-adaptation in task scheduling. In this configuration, tasks are processed sequentially as they are received, with no particular priority assigned to any task, thereby treating all tasks with equal importance. The sliders in Figure 5(b) can be used by the user to assign weights to the emotions. Subsequently, these weights are applied as multipliers to the sentiment predictions generated by the NLP model, thereby enhancing the priority of the desired sentiment. As outlined in the methodology section, in cases of repeated instructions expressing the same emotion, the system dynamically adjusts the weights assigned to various emotions to prioritize the user's current emotional state. For instance, if a user has consistently expressed anger over a recent period, the system intelligently increases the weighting for the sad emotion. For the sake of simplicity, all negative emotions are collectively categorized as 'sad,' while positive emotions fall under the category of 'happy.' This adaptive approach ensures that the system not only responds to the user's emotional cues but also adapts its weighting scheme to align more closely with the user's prevailing emotional context.

In contrast, the system's capability to prioritize tasks based on specific criteria is highlighted through a different approach. This is evident in the ordering that accounts for factors such as health emergencies, sensitivity to health-related issues, and the emotional context of each command. This prioritization mechanism is visually detailed in Figure 6, where we showcase the various pieces of information extracted from each instruction. These data points play an indispensable role in the system's calculation of task priority, enabling tasks to be processed with more than just a sequential approach. Instead, the system demonstrates a nuanced understanding of their varying levels of importance and urgency. By delving into the system design elucidated in the methodology section, we gain insights into how different aspects contribute to the priority score. Notably, health emergency instructions are accorded the highest priority, with other commands assigned distinct priorities depending on their contextual relevance and



(a) (b)

Fig. 7. Inference graphs for long-term adaptation (a) Average emotions throughout the day, (b) Objects to focus at different times of the day based on the negative emotion scores.

emotion-based weighting. This systematic approach results in a more responsive and adaptable system that aligns better with the user's needs and emotional state, enhancing the overall user experience.

C. Inference from user instructions

In this section, we will discuss how future implementations can infer from the user data to adapt better. The data depicted in Figure 7(a) represents the 5-day average sentiment scores categorized as happy, sad, and neutral, recorded over different times of the day. These scores were derived from an analysis of user texts over 5 days using a sentiment analyzer module. We can infer from this plot that the average sentiment score for 'Happy' emotions peaks in the afternoon, suggesting that users tend to express more positive sentiments during this time of the day. There is a noticeable decline in 'Happy' sentiment as the day progresses into the night. On the other hand, 'Sad' sentiment scores are lowest in the afternoon and show a gradual increase, peaking in the night. This could indicate that users are more likely to express sadness or negative emotions towards the end of the day. It is also observable that the user exhibits higher levels of sadness and lower levels of happiness in the morning as opposed to the afternoon and evening. The 'Neutral' sentiment remains relatively stable throughout the day, with slight fluctuations that do not follow a clear trend. The data from a five-day sentiment analysis suggests that users' moods are generally more positive in the morning and afternoon, then shift toward more negative sentiments by night. The 'Neutral' sentiment is consistent across all times, suggesting that the most of user's commands are typically routine tasks without emotional aspects. These patterns could inform targeted strategies to improve user experience at different times of the day.

Bridging the understanding between the overall emotional trend and the objects of focus at different times of the day is shown in Figure 7(b). We wanted to infer different objects that the user can seek at different times of the day with a negative emotion, indicating the robot to be more attentive over time to improve the Human-Robot Interaction. This depth enriches the broader patterns observed in Figure 7(a) by providing context to the fluctuations in sad sentiment scores. This detailed analysis from Figure7(b) gives substance to the emotion patterns depicted in Figure 7(a), painting a fuller picture of the user's daily emotional journey.

The analysis of the average sadness score over different times of the day, as illustrated in Figure 7, reveals a significant trend in the user's emotional state. Throughout the day, the top three sad keywords associated with each period offer insight into the reasons behind the user's emotional state. In the morning, with "keys", "dirty," and "diabetes" as the primary sad keywords, the user begins the day with moderate sadness. These keywords suggest potential frustration with cleanliness, and the urgency to find keys or issues related to health. The afternoon shows a significant decrease in sadness, with context keywords such as "schedule," "monitor," and "garden." This suggests that while there is some discontent possibly related to time management like meetings, supervision of tasks, or gardening, it is less intense than in the morning, and most of the user tasks seem to make the user calm. However, as the day progresses into the evening, the sadness score surges, with "arrange," "room," and "mood" being the most significant concerns. This substantial increase could reflect the emotional toll of organizing personal spaces or a general deterioration in mood as the day comes to a close. However, the night shows the highest sadness score, with "bills," "car," and "maintenance" as the top sad keywords. These issues, likely related to financial obligations and personal maintenance tasks, represent the most significant emotional challenges for the user. This presence of specific keywords provides a concrete link to potential stress points in the user's day and opportunities for the robot to adapt and plan strategies.

IV. Conclusions

Our project was primarily focused on addressing the challenge of enhancing Human-Robot Interaction (HRI) through an advanced self-adaptive software system. The core objective was to develop a solution that transcends traditional sequential task scheduling, integrating a more nuanced understanding of context, including sentiment, time, and health sensitivity. These factors are crucial as they mirror the decision-making process humans employ in daily life. Our solution, tailored for an indoor home environment, innovatively combines principles from Operating Systems with the architectural framework of MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) self-adaptive software systems. This integration led to the development of a preemptive priority scheduling system, where priorities are dynamically determined using MAPE-K concepts. This represents a forward leap in the field of intelligent assistive agents, offering a more responsive and context-aware approach to task management.

Throughout the project, our learning journey was extensive and multifaceted. We delved into the realms of Self-Adaptive Software Systems, models for Software Development, Natural Language Processing, Artificial Intelligence, Operating Systems, Web Application Development, Human-Robot Interac-

tion, and Cognitive Sciences. The synthesis of these diverse areas was instrumental in forging our innovative solution. We learned the importance of interlinking various concepts to create a cohesive and effective system. The structured frameworks of MAPE-K and the three-layer architecture model proved invaluable. They allowed us to compartmentalize complex topics, addressing each segment individually before integrating them into a unified whole. This approach not only streamlined our development process but also enhanced our understanding of each component's role within the larger system.

Furthermore, the project underscored the effectiveness of the incremental model in software development. This approach, which advocates for starting with a basic implementation and progressively adding features, was pivotal in managing the complexity of our project. It enabled us to focus on core functionalities initially and then expand our system's capabilities in a structured manner.

In this report, we have endeavoured to contribute positively to the field of Human-Robot Interaction (HRI), a domain that requires an intricate blend of engineering and an understanding of human psychology. Both these areas are highly dynamic, necessitating innovative approaches like the application of self-adaptive software systems. Despite our efforts, there were several limitations to our implementation, which we highlight below:

- **Decision-Making Factors:** The focus was primarily on emotions, time, and health sensitivity, overlooking factors like historical interaction context and personal preferences.
- **Interface Constraints:** The text-based interface simplifies interactions but doesn't capture the complexities of multimodal communication in HRI scenarios.
- **Task Prioritization:** The system's task prioritization may not fully address the subjective nature of situational context and individual user preferences.
- **Real-World Deployment:** The study was limited to interface development without testing in a real-world robotic system, potentially overlooking practical complexities.
- **Interface Simplicity:** The interface, designed for single-context instructions, lacks advanced dialogue capabilities, limiting its effectiveness in complex interactions.

In conclusion, this project represents not only a step forward in the realm of Human-Robot Interaction (HRI) but also embodies a journey that amalgamated various fields of study. It highlights the art of interdisciplinary integration and the power of incremental progress in tackling complex, multi-faceted problems. While our research has made significant strides, acknowledging the limitations encountered is crucial. This introspection is essential for guiding future research and development efforts, ensuring continuous advancement in this rapidly evolving field.

V. Future Work

In our future work, we aim to significantly enhance our system by integrating multimodal inputs, including audio and video. This advancement will enable the system to not only record instructions more effectively but also to capture and interpret emotional cues. Following an incremental model of software development, enabling continuous refinement and evolution of the system to progressively enhance its capability to mimic human decision-making processes. This will involve progressively incorporating more complex aspects of human cognition and prioritization into the robot's task management algorithms. Once the enhanced functionalities are established and operating smoothly, our next step will be to deploy the software in a robotic platform.

This deployment will mark a pivotal transition to actual Human-Robot Interaction (HRI) experiments. We plan to conduct comprehensive studies with participants to assess how these advancements impact the perception of the robot's intelligence and likeability. Such interactions are crucial for understanding and improving the dynamics of human-robot relationships.

Our ultimate goal is to create a robot that not only performs tasks efficiently but also resonates well with human emotions and social norms, thereby enhancing the overall user experience and acceptance of robotic assistants in everyday life.

Acknowledgment

We express our sincere gratitude to Professor Ladan Tahvildari and our teaching assistants, Mingyang Xu and Ryan Zheng He Liu, for their invaluable guidance throughout our course. Our thanks also extend to the IBM team for providing resources that were crucial in applying our skills to real-world systems and enhancing our understanding of MAPE-K based self-adaptive software systems. The assignments and quizzes significantly aided in translating theoretical knowledge into practical solutions, and we are deeply appreciative of the support and opportunities offered by our instructors, which have been instrumental in our academic growth.

References

[1] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "Cobots: Robust symbiotic autonomous mobile service robots," in *Twenty-fourth international joint conference on artificial intelligence*. Citeseer, 2015.

[2] A. Silberschatz and P. B. Galvin, "Operatings system concepts," 2013.

[3] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[4] J. Hartmann, "Emotion english distilroberta-base," https://huggingface.co/j-hartmann/emotion-english-distilroberta-base/, 2022.

[5] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *ArXiv*, vol. abs/1910.01108, 2019.

[6] F. AI, "Bart large mnli," https://huggingface.co/facebook/bart-large-mnli, 2022, accessed: 2023-12-02.

[7] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019.

[8] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," 2018.

[9] D. Weyns, *An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*, 1st ed. Wiley-IEEE Press, 2020, ch. 5.

[10] D. Graham, "Incremental development: review of nonmonolithic life-cycle development models," *Information and Software Technology*, vol. 31, no. 1, pp. 7–20, 1989.