Human-in-the-loop Foundation Model Failure Recovery for Robot-Assisted Bite Acquisition

Krishna Palempalli[1], Rohan Banerjee[1], Sarah Dean[†1], Tapomayukh Bhattacharjee[†1]

***Abstract*— Robot-assisted bite acquisition requires accurately identifying food items and selecting the correct high-level action (e.g., skewering, scooping, or twirling) to acquire them. While foundation models such as Large Language Models (LLMs) and Vision Language Models (VLMs) enable modular and generalizable perception-to-action pipelines, their deployment can result in occasional failures—particularly when faced with diverse or ambiguous food items. To ensure safety and reliability in such settings, we propose selectively querying the care recipient for help when uncertainty arises. However, excessive or poorly timed queries can impose a workload on users, especially those with mobility limitations. We introduce a modular human-in-the-loop framework that queries across different components of the pipeline using querying rules informed by both model uncertainty and predicted user workload. We define three querying rules and evaluate them in offline simulations using realistic food plate images. Our results show that all querying rules improve task performance over the baseline, with each offering a different trade-off between task performance and query efficiency. Our framework generalizes beyond assistive feeding and provides a principled approach for safe, efficient querying in foundation model-driven robotics systems.**

I. INTRODUCTION

Large Language Models (LLMs) and Vision Language Models (VLMs) [1]–[5] are increasingly used in robotics for perception, task planning, and skill execution. While these foundation models provide generalizable capabilities, their integration into robotics systems—for instance, in robot-assisted feeding [6], where a robot needs to feed a care recipient with mobility limitations—raises critical concerns around safety and reliability. For the problem of robot-assisted bite acquisition [6]–[8], in which a robot picks up food items from a plate, safety could entail the robot correctly identifying a food item, and selecting the right acquisition action. However, foundation model failures for this task may manifest at multiple stages, from perception (such as misclassification or object detection) to action selection, because of the diversity of food items that exist in the wild. Repeated bite acquisition failures could unsafely impact the care recipient's health and nutrition.

Identifying failures and autonomously recovering from them is challenging. Each module—perception or action—may fail for different reasons and require different recovery strategies. A natural solution in assistive settings is to involve the care recipient by allowing the robot to query them for help [9, 10]. Prior work has explored querying strategies based on post-failure feedback [11, 12], or based on task uncertainty measures such as action confidence estimates [13], [14] and expected information gain [15, 16]. Feedback types also vary widely, from clarifying goals [13] to providing labels, explanations, or demonstrations [11, 12, 17].

Our work differs from this prior work in two key respects. First, frequent or ill-timed queries can introduce cognitive and physical workload, especially for users with mobility limitations [18, 19]. A safe assistive robot must responsibly query the human, as unnecessary queries could lead to frustration and potentially even incorrect feedback. Thus, improving task performance with minimal user workload is essential for any safe human-in-the-loop assistive system. Second, while most prior approaches focus on querying a single module, we propose a modular querying framework that targets different components of a perception-to-action pipeline, and reasons jointly using model confidence and workload estimates to make safe query decisions.

In this work, we focus on the task of bite acquisition, where a robot needs to acquire food items from a plate. Our modular pipeline includes two GPT-4o-based VLMs [20]: a perception module for food item identification, and an action-selection module for high-level acquisition action selection (e.g., skewering, scooping, twirling). Each module produces candidate options along with associated confidence scores. In our setting, queries to the user differ in type (food items vs. high-level actions) and in the associated workload costs.

To decide when to query, one could directly use raw confidence scores that come from the foundation models. However, these scores are often miscalibrated and unstable, with LLMs frequently exhibiting overconfidence or underconfidence in their predictions [21, 22]. This miscalibration can be especially problematic in out-of-distribution scenarios, underscoring the need for more reliable uncertainty estimation methods. Moreover, recent work also highlights the broader importance of quantifying uncertainty in language models to support safer, more reliable decision-making [23]. To address this, we developed a suite of querying rules designed to produce more reliable querying decisions based on the model's confidence in its prediction, estimated success rate of the prediction, and estimated user workload. Through offline simulations on a dataset of real-world food plates, we evaluate these rules and show that our modular, uncertainty-aware querying framework improves safety and accuracy for the robot's bite acquisition behavior, with minimal querying. While our paper focuses on the assistive feeding domain, the underlying human-in-the-loop framework applies broadly across robotics domains to effectively improve task performance and safety.

[†]Equal Advising.

[1]Department of Computer Science, Cornell University {kp386, rbb242, sdean, tapomayukh}@cornell.edu

II. PROBLEM FORMULATION

We explore the task of bite acquisition in a setting where plates of food items contain multiple instances of various food types from a predefined set $\mathcal{F}$. The goal is to successfully acquire every instance of each food type present on the plate. We assume the bite-acquisition robot operates with an observation space $\mathcal{O}$ (RGB images) and a high-level action space $\mathcal{A}_h$ containing three discrete actions: skewering, scooping, and twirling [6]. Additionally, we assume the presence of a modular base architecture for performing bite acquisition, which defines a policy $\pi : \mathcal{O} \to \mathcal{A}_h$, structured as a sequence of N modules $m_1, m_2, \ldots, m_N$. Each module m_i produces a specific output along with confidence scores for the output c_i, resulting in $\mathcal{C} = [c_1, c_2, \ldots, c_N] \in [0, 1]^N$. Every module m_i is called exactly once when making a policy decision for a given input observation $z_{\text{RGB}} \in \mathcal{O}$. Now in our setting, we consider two modules ($N = 2$), a perception module and an action-selection module.

- $\mathbf{m_1}$: food item detector with GPT-4o LLM + VLM capabilities [20]. This module processes an RGB image, $z_{\text{RGB}} \in \mathbb{R}^{H \times W \times 3}$, of a whole plate and detects K, the number of unique food item categories (e.g., lettuce, watermelon, etc.) on the plate, and identifies a set of food labels $\mathcal{L} = \{l_1, l_2, \ldots, l_K\}$. For each food item, the label is the output token with the highest probability (confidence score). From $\mathcal{L}$, we then arbitrarily select a label l_i for acquisition.
- $\mathbf{m_2}$: high-level action selector with GPT-4o LLM + VLM capabilities [20]. Given a detected and selected food item label l_i and the corresponding RGB image, $z_{\text{RGB}} \in \mathbb{R}^{H \times W \times 3}$ of the whole plate, this module predicts the optimal high-level action $a_i^h \in \mathcal{A}_h$ for that food item. The high-level action a_i^h is the output token (action) with the highest probability (confidence score).

With this setting, we assume that the dishes can be in configurations that can cause the base policy π to fail, resulting from a failure in at least one module m_i. For each module, the robot can ask the human a corresponding query $q_i \in \mathcal{Q}$ and the human provides expert feedback f_i in response to the query (e.g. correct food item label l_i for m_1 or correct high-level action a_i^h for m_2. Each query type q_i also incurs a corresponding querying workload w, which accounts for both cognitive and physical workload relevant for users with mobility limitations [18]. We assume that the system has access to a predictive workload model for estimating the workload w[1] if we were to ask the human a particular type of query q_i at a particular timestep, taking into account the human's initial querying workload and the history of queries [24].

Using all of this, our goal is to learn a failure recovery policy $\pi_f : (\mathcal{C}, \mathcal{A}_h) \to (\mathcal{A}_h \cup P(\mathcal{Q}))$, which either decides to execute the autonomous high-level action a_h or opt to ask the human a selected subset of queries $q \subseteq \mathcal{Q}$.

[1] w depends on the current timestep t, but to maintain brevity in the paper, we remove t from the notation for this term.

III. QUERY METHODOLOGY

A. Querying Setup

To enable uncertainty-aware querying, we construct calibration datasets and associated confidence or success-rate intervals for both modules in our bite-acquisition pipeline: the food item detector (m_1) and the high-level action selector (m_2). The three querying rules (introduced in Section III-B) depend on different types of calibration data and intervals because two of the rules query on the basis of confidence scores and their variances, and one rule queries on the basis of predicted success rate and its variance.

Calibration dataset construction. The confidence-based querying rules rely on instance-independent confidence intervals built from calibration data specific to each module. Each calibration dataset consists of per-instance model outputs (tokens and confidence scores) along with corresponding ground truth labels, extracted from plate images.

For module m_1, we use the dishes and plate images from [6]. For each image, we treat every food item present in the image as a target food item and run m_1 to generate tokens (potential labels) for each food item, along with the log probability score for each token. The log probabilities are exponentiated, multiplied by 100, and rounded to two decimal places to produce percentage values representing the confidence scores. For each food item, the top token with the highest confidence score is selected as its label.

For each target food item, if the target food item is among the identified predictions (or equivalently, if the top token selected for that item matches the item's ground truth label), we treat the item as correctly identified. In this case, we add an entry containing the top 5 tokens, their confidence scores, and the ground truth label to the calibration dataset corresponding to that food item and image pair. If the target food item is not among the identified predictions, we randomly choose a substitute from the remaining unmatched predictions for that image (if any remain). If no unmatched predictions remain, we reuse a matched prediction from the same image as a fallback. In either case, we add the corresponding information (the top 5 tokens, their confidence scores, and the ground truth label) of the substitute to the calibration data, representing the data entry for the target food item and image pair. This ensures every target item—identified or not—has a corresponding calibration entry, allowing us to later compute the necessary intervals.

For module m_2, we use the outputs of m_1 to generate the inputs for each target food item and image pair. The identified (or substituted) food label, along with the image, is passed to m_2, which generates potential action tokens along with their log probability scores. These are converted to confidence scores using the same process as for m_1, and the top token with the highest confidence score is selected as the high-level action for that instance. We add the top $|\mathcal{A}_h|$ tokens, their confidence scores, and the ground truth high-level action to our calibration data corresponding to that target food item and image pair. This results in a confidence-based calibration dataset for m_2 similar to that of m_1.

Confidence and success rate intervals. Using the cor-

responding calibration datasets, for modules m_1 and m_2, we compute the mean and standard deviation of the top choice (token with the highest confidence score) confidence scores and the second top choice (token with the second highest confidence score) confidence scores across all the entries in the calibration data, denoted by μ^*, σ^* and μ^+, σ^+, respectively. These are used to define the following instance-independent confidence intervals—top choice confidence interval I_{TopConf} and second top choice confidence interval $I_{\text{SecondTopConf}}$—for each module:

$$I_{\text{TopConf}}: \quad [\mu^* - \sigma^*,\ \mu^* + \sigma^*]$$
$$I_{\text{SecondTopConf}}: \quad [\mu^+ - \sigma^+,\ \mu^+ + \sigma^+]$$

The computed values for module m_1 are $\mu^* = 93.62$, $\sigma^* = 12.51$, $\mu^+ = 5.82$, and $\sigma^+ = 11.66$. For module m_2, we obtain $\mu^* = 93.64$, $\sigma^* = 13.72$, $\mu^+ = 5.24$, and $\sigma^+ = 11.20$. The above intervals and values are used by the two confidence-based rules as described in Section III-B. Figure 1 (left) provides an overview of how the calibration dataset and confidence intervals are constructed for m_2.

In contrast to the confidence-based rules that depend on precomputed calibration data and instance-independent confidence intervals, the success rate rule relies on instance-specific intervals that do not rely on a precomputed calibration dataset. For each input instance (the target food item and image pair), we run the corresponding module for $N_{\text{batch}} = 10$ trials. We then identify the majority token, which is the token that appears as the top choice across the majority of trials. For each of the 10 trials, we label the top choice and second top choice as successes if each matches with the majority token. We compute the batch's success rate s^* (proportion of successes across the 10 trials) for the top choice, along with the standard deviation σ^* of these successes and the batch's success rate s^+ (proportion of successes across the 10 trials) for the second top choice, along with the standard deviation σ^+ of these successes[2]. These are used to define instance-specific success rate intervals—top choice success rate interval I_{TopSucc} and second top choice success rate interval $I_{\text{SecondTopSucc}}$—for each module:

$$I_{\text{TopSucc}}: [s^* - \sigma^*,\ s^* + \sigma^*]$$
$$I_{\text{SecondTopSucc}}: [s^+ - \sigma^+,\ s^+ + \sigma^+]$$

The above intervals are used by the success rate rule described in Section III-B. Figure 1 (right) provides an overview of how the binary success labels and instance-specific success rate intervals are constructed for m_2.

This overall querying setup provides the foundation for defining the rules described in the next section, allowing the system to reason jointly over model uncertainty and user workload in a structured and interpretable manner.

B. Querying Rules

We describe the three querying rules for both modules m_1 and m_2 in our bite-acquisition architecture (illustrated in Figure 2 (left)), defining the condition do_query under which each rule queries. To mitigate the unreliability of using raw probability values directly as mentioned before, we design rules that measure confidence and uncertainty in more stable, interpretable ways. The two confidence-based rules are denoted *Rule 1a* and *Rule 1b*, and the success rate-based rule is denoted *Rule 2*. For both m_1 and m_2, depending on the rule, we use the respective confidence scores (c_i) for the top choices, as described in Section II, or the respective success rates for the top and second top choices (s^* and s^+), as described in Section III-A. In addition, the rules rely on the predicted querying workload w at each timestep for each potential query, as described in Section II. To decide when to query the human, the rules use both the confidence scores or success rates, and the workload estimates.

Rule 1a. We introduce c^* to represent the confidence score of the top choice in general (either for the food label or high-level action). This rule allows us to query when c^* deviates from the norm, helping us catch outlier predictions that are either under-confident or over-confident. Specifically, for the current instance, we decide to query if its c^* falls outside the calibrated I_{TopConf}, or overlaps with the calibrated $I_{\text{SecondTopConf}}$, provided that the predicted workload is below ϵ, the querying threshold for the workload:

$$do\_query = (c^* \notin I_{\text{TopConf}} \ \vee \ c^* \in I_{\text{SecondTopConf}}) \wedge (w < \epsilon).$$

This rule constructs a more stable and interpretable uncertainty measure by examining the model's confidence in relation to typical distributions observed during calibration. Specifically, we compute the confidence range for both the top and second top predictions across all food items of interest (I_{TopConf} and $I_{\text{SecondTopConf}}$), treating these as population-level reference intervals. By comparing an instance's confidence to these intervals, we identify cases of overconfidence (where a low-quality prediction appears unusually confident) or underconfidence (where a correct prediction appears uncertain). This population-informed approach filters out unreliable predictions that deviate from the norm.

Rule 1b. This is a modified version of Rule 1a, which uses a batch of repeated trials on the same instance, instead of just a single trial, to mitigate stochasticity in the LLM/VLM outputs. For the current instance, we run the relevant module for $N_{\text{batch}} = 10$ trials, and compute the average of the top choice confidence scores c^* across the batch to obtain the mean top choice confidence score for the batch, denoted $\bar{c^*}$. In contrast to Rule 1a, Rule 1b uses $\bar{c^*}$ instead of c^*[3]:

$$do\_query = (\bar{c^*} \notin I_{\text{TopConf}} \ \vee \ \bar{c^*} \in I_{\text{SecondTopConf}}) \wedge (w < \epsilon).$$

Rule 1b extends Rule 1a by using batch-averaging to mitigate the stochasticity of individual forward passes. By repeating the model's prediction multiple times for each instance, we compute a more stable, lower-variance top confidence interval I_{TopConf}.

[2] s^*, s^+, σ^*, σ^+ all depend on the current timestep t, but to maintain brevity in the paper, we remove t from the notation for these terms.

[3] c^* and $\bar{c^*}$ depend on the current timestep t, but to maintain brevity in the paper, we remove t from the notation for this term.

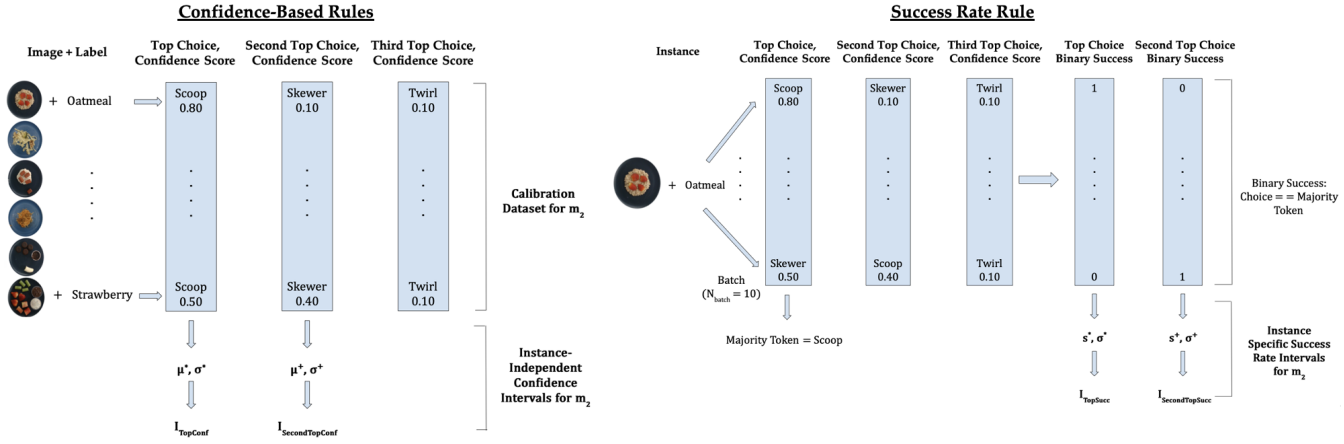

Fig. 1: Overview of the calibration and uncertainty estimation process for the high-level action selector module m_2. (left) Illustration of calibration dataset construction for m_2. For each image and identified food label, we extract and store the all predictions (and their confidence scores) in the calibration dataset. These scores are used to compute the instance-independent confidence intervals I_{TopConf} and $I_{\text{SecondTopConf}}$. (right) Illustration of success rate interval construction, where we run the model $N_{\text{batch}} = 10$ times on the same instance to obtain all predictions (and their confidence scores) for the N_{batch} trials. We select a majority token across the top choices, and we score the trial-specific top and second-top choices as binary successes if they match the majority token. We use these to compute the instance-specific success rate intervals I_{TopSucc} and $I_{\text{SecondTopSucc}}$.

Rule 2. This rule uses the overlap between the two success rate intervals—I_{TopSucc} and $I_{\text{SecondTopSucc}}$—as a proxy for prediction ambiguity, and queries if the second top choice appears comparably successful to the top choice under the model's uncertainty. Specifically, for the current instance, we decide to query if the gap value G, measuring the overlap between I_{TopSucc} and $I_{\text{SecondTopSucc}}$ (either for the food label or high-level action), is greater than the scaled workload [24]:

$$do\_query = G > w \cdot \epsilon$$

where

$$G = (s^+ + \sigma^+) - (s^* - \sigma^*).$$

Unlike Rules 1a and 1b, which rely on population-wide calibration statistics, Rule 2 addresses the model's own internal uncertainty. We estimate this by measuring how consistent the model is across multiple forward passes. If the same top choice is selected repeatedly, the model is epistemically confident in its output; if not, the model is uncertain. In this way, we capture both overconfidence (when the model varies but one prediction dominates) and underconfidence (when predictions vary widely with no dominant choice). This method provides a robust, instance-specific uncertainty estimate that adapts to unfamiliar or ambiguous inputs.

IV. EXPERIMENTS

A. Simulation Setup

Calibration and test dishes. We use a set of images from six realistic, in-the-wild calibration dishes to tune and select the querying rules' hyperparameters for modules m_1 and m_2. These calibration dishes with a variety of visual and material characteristics are shown in Figure 2 (right, top) [6]:

- A plate with strawberries and oatmeal.
- A plate with sausage pieces and mashed potatoes.
- A plate with meatballs and spaghetti noodles.
- A plate with fettucine noodles and chicken and broccoli pieces.
- A plate with brownies and sliced banana.
- A plate with strawberries, cantaloupe pieces, and celery pieces.

We then assess the performance of our querying rules on four realistic, in-the-wild test dishes, each containing food items with diverse visual and material properties, as illustrated in Figure 2 (right, bottom):

- A fruit salad with bite-sized fruits: watermelon, sliced banana, cantaloupe, honeydew, grapes, blueberries.
- A savory salad: lettuce, chicken pieces, cherry tomatoes.
- A Thanksgiving meal: chicken pieces, mashed potatoes, green beans.
- A noodle plate: Fettucine noodles, chicken pieces, cherry tomatoes.

Metrics. We define the following evaluation metrics for each querying rule: true positive (TP) cases of when the rule decides to query, and the output from the module is incorrect, true negative (TN) cases of when the rule decides not to query, and the output from the module is correct, false positive (FP) cases of when the rule decides to query despite the module's output being correct, and false negative (FN) cases of when the rule decides not to query despite the module's output being incorrect. Along with these, we also use the total number of queries N_q encountered, the normalized Matthews Correlation Coefficient ($nMCC$) [25], and the overall success rate (SR), which is measured as the total number of successful food item acquisitions divided by the total number of timesteps taken to complete the simulation.

Simulation description. We evaluate the three querying rules for modules m_1 and m_2 using offline plate image data. The simulation assumes access to ground-truth food item labels and their corresponding high-level actions (a_h). For

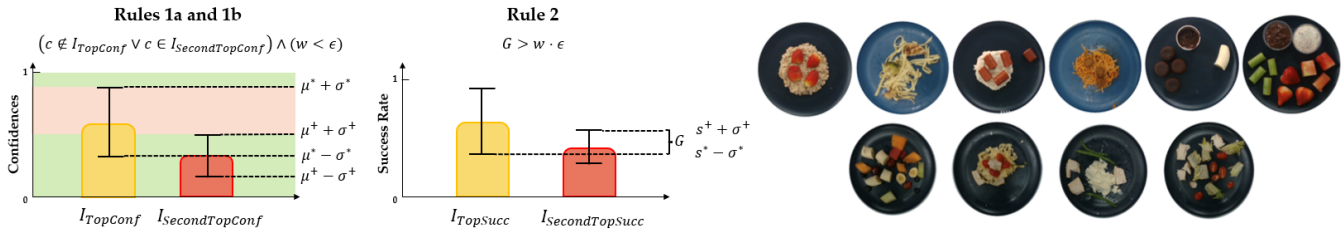

Fig. 2: (left) Illustration of the foundation model querying rules. For Rules 1a and 1b, we query if the confidence score c (top choice confidence c^* for 1a, mean top choice confidence $\bar{c^*}$ for 1b) lies within either of the green regions, and we don't query if c lies within the red region. For Rule 2, we query if $G > w \cdot \epsilon$. (right, top) Calibration dishes used for querying rule hyperparameter selection [6], (right, bottom) Test dishes used for querying rule evaluation.

each dish in the calibration set, a sequence of images is processed, where the sequence simulates one food item being acquired per image until all items are acquired from the plate.

For Rule 1a, the two modules are run in order on the corresponding image for each food item, representing a single discrete timestep. The module's output is considered a success if it matches the ground-truth; otherwise, it is considered as a failure. For Rules 1b and 2, the two modules are run in order on the corresponding image for each food item for $N_{\text{batch}} = 10$ trials, still representing a single discrete timestep. If the most frequently occurring top choice token across the 10 trials matches the ground-truth, it is considered as a success; otherwise, it is considered as a failure. Regardless of the output, at each timestep, the querying rule is applied to decide whether a query should be made. If the rule says we should query, then the instance is treated as a success. If we don't query and have a failure for that instance, then the two modules are rerun in order on the instance. This process is repeated for up to $T = 5$ timesteps. If no success is encountered within those $T = 5$ timesteps, the instance is treated as a failure, and the simulation proceeds to the next food item and corresponding image in the sequence.

This simulation is repeated for 10 different ϵ values, ranging from 0.05 to 0.5 in increments of 0.05. The ϵ value resulting in the highest $nMCC$ and SR values is selected as the best value and is used to evaluate the querying rules on the test dishes using the same simulation setup. This entire process is repeated for each of the three querying rules for the two modules m_1 and m_2.

B. Results

Tables I and II show the metrics and results of the simulation on the offline data. The baseline (no-querying) performs significantly worse, especially on the test dishes. For example, the baseline achieves only 0.42 SR on m_1 and 0.38 SR on m_2, revealing that without querying, the system fails to recover from model errors. Interestingly, the $nMCC$ values for the baseline remain flat at 0.50 across all cases, showing that while the system is partially consistent in its predictions, it is not accurate enough to guarantee successful bite acquisition without human input. This limitation becomes even clearer when we observe that the baseline produces zero TP cases and a large number of FN cases, 18 and 20 for m_1 and m_2, respectively, confirming that the system often misses opportunities to correct its own failures when it doesn't query. Although FP cases are low at 0, this comes at the cost of missing nearly all correctable mistakes. So the abundance of FN cases drags both SR and $nMCC$ down, leading to the weak performance of the baseline.

For both modules m_1 and m_2, all three querying rules generally perform better than the baseline, with only a small number of queries. For module m_1, Rule 1a, which relies on instance-independent confidence intervals, achieves the highest success rate (0.64) and a strong $nMCC$ (0.48) with only 5 queries on the 16 test food items, showing strong predictive performance. Importantly, this improvement is supported by a meaningful shift in the confusion matrix: Rule 1a recovers 2 TP cases and reduces FN cases to 9, compared to 18 in the baseline, showing that even a small number of well-placed queries can substantially increase the system's success rate. The cost is a modest increase in FP cases to 3, which leads to a slightly lower $nMCC$ for Rule 1a (0.48) compared to the baseline (0.50). However, the gains in both TP cases and SR and the drop in FN cases outweigh the slight decrease in $nMCC$. Rule 1b performs similarly to Rule 1a, suggesting robustness in performance across instance-independent confidence-based rules.

We find that Rule 2, which relies on instance-specific success rate intervals, queries only twice in m_1 on the 16 test food items, yet still recovers 2 TP cases while eliminating FP cases entirely and reducing FN cases to 10. These lead to Rule 2 achieving a higher SR (0.58) than the baseline and attaining the best $nMCC$ (0.65) for m_1, demonstrating strong prediction reliability for the instance-dependent success rate rule.

For module m_2, the overall pattern remains consistent. Rule 1a again achieves the highest SR (0.58) and the best $nMCC$ (0.65), with 2 TP cases, 10 FN cases, and no FP cases, using only 2 queries on the test 16 test food items. Also, Rule 1b's performance matches with Rule 1a's performance, showing strong predictive performance. Rule 2 is slightly more conservative as it queries only once, achieving slightly lower $nMCC$ (0.58) and SR (0.46) values than Rules 1a and 1b as Rule 2 recovers slightly fewer TP cases and has slightly higher FN cases. But the SR, $nMCC$, and TP values are still higher than the baseline's values, and the FN value is still lower than the baseline's value. These results illustrate that all three rules offer substantial improvements over the baseline, but they differ in how they

		Calibration	Test
Rules	Best ϵ	$FP\downarrow, FN\downarrow, TP\uparrow, TN\uparrow$	$FP\downarrow, FN\downarrow, TP\uparrow, TN\uparrow$
BASELINE	-	$0, 0, 0, 14$	$0, 18, 0, 13$
RULE 1A	0.40	$0, 0, 0, 14$	$3, 9, 2, 11$
RULE 1B	0.40	$0, 0, 0, 14$	$2, 10, 2, 10$
RULE 2	0.05	$0, 0, 0, 14$	$0, 10, 2, 12$
		Calibration	Test
Rules	Best ϵ	$FP\downarrow, FN\downarrow, TP\uparrow, TN\uparrow$	$FP\downarrow, FN\downarrow, TP\uparrow, TN\uparrow$
BASELINE	-	$0, 20, 0, 10$	$0, 20, 0, 12$
RULE 1A	0.45	$0, 20, 1, 9$	$0, 10, 2, 12$
RULE 1B	0.35	$0, 20, 2, 8$	$0, 10, 2, 12$
RULE 2	0.05	$0, 20, 1, 9$	$0, 15, 1, 12$

TABLE I: Simulation results, showing FP, FN, TP, and TN, for all three querying rules along with the baseline (no querying at all) on offline images from the calibration and test dishes for the food item detector module m_1 (top table) and high-level action selector module m_2 (bottom table).

		Calibration	Test
Rules	Best ϵ	$N_q\downarrow, nMCC\uparrow, SR\uparrow$	$N_q\downarrow, nMCC\uparrow, SR\uparrow$
BASELINE	-	$0, 0.50, 1.00$	$0, 0.50, 0.42$
RULE 1A	0.40	$0, 0.50, 1.00$	$5, 0.48, 0.64$
RULE 1B	0.40	$0, 0.50, 1.00$	$4, 0.50, 0.58$
RULE 2	0.05	$0, 0.50, 1.00$	$2, 0.65, 0.58$
		Calibration	Test
Rules	Best ϵ	$N_q\downarrow, nMCC\uparrow, SR\uparrow$	$N_q\downarrow, nMCC\uparrow, SR\uparrow$
BASELINE	-	$0, 0.50, 0.33$	$0, 0.50, 0.38$
RULE 1A	0.45	$1, 0.56, 0.33$	$2, 0.65, 0.58$
RULE 1B	0.35	$2, 0.58, 0.33$	$2, 0.65, 0.58$
RULE 2	0.05	$1, 0.56, 0.33$	$1, 0.58, 0.46$

TABLE II: Simulation results, showing N_q, $nMCC$, and SR, for all three querying rules along with the baseline (no querying at all) on offline images from the calibration and test dishes for the food item detector module m_1 (top table) and high-level action selector module m_2 (bottom table).

trade off between task performance and human workload.

Rather than framing one rule as definitively superior, our findings show that the suite of rules offers different trade-offs depending on system design goals. If maximizing the average test success rate across the modules is prioritized, then Rule 1—and more generally, instance-independent rules that use calibrated confidence intervals—might offer stronger task performance with a reasonable number of queries as they have fewer FN cases and recover several TP cases. If minimizing the number of queries, while still achieving reasonable performance, or maximizing average test $nMCC$ across the modules is more important, instance-dependent rules that use success rate intervals like Rule 2, may be preferred for their stability and lower workload, with a low number of FP cases and recover a few TP cases.

In this way, the querying rules form a generalizable toolkit for safe and efficient human-in-the-loop decision-making in LLM and VLM based systems. Our results suggest that safety-critical LLM/VLM applications may benefit from selecting rules aligned with the desired trade-off between user workload and task success. Across both modules, all rules improve task performance, while requiring only a handful of queries, demonstrating the value of incorporating uncertainty and workload estimates into the decision-making. One promising future direction would be to incorporate learning from the human feedback, where we use techniques such as retrieval-augmented generation (RAG) [26] to improve the foundation model's performance on future food items, thus further improving system safety.

Overall, our modular querying framework contributes to the goal of building trustworthy, human-aligned robotic systems. By leveraging foundation model uncertainties and predicted user workload, we show how robots can intervene only when necessary—achieving safer, more accurate behavior with minimal user disruption. These querying strategies are not only effective in assistive feeding, but can be generalized to other foundation model-driven robotics pipelines.

ACKNOWLEDGMENT

This work was partly funded by NSF CCF 2312774 and NSF OAC-2311521, a LinkedIn Research Award, and a gift from Wayfair, and by NSF IIS 2132846 and CAREER 2238792. Research reported in this publication was additionally supported by the Eunice Kennedy Shriver National Institute Of Child Health & Human Development of the National Institutes of Health and the Office of the Director of the National Institutes of Health under Award Number T32HD113301. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. The authors would like to thank Yuanchen Ju for helping with figure creation.

REFERENCES

[1] S. Karamcheti, S. Nair, A. S. Chen, *et al.*, "Language-driven representation learning for robotics," *arXiv preprint arXiv:2302.12766*, 2023.

[2] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PmLR, 2021, pp. 8748–8763.

[3] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] A. Chowdhery, S. Narang, J. Devlin, *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[5] D. Driess, F. Xia, M. S. Sajjadi, *et al.*, "Palm-e: An embodied multimodal language model," 2023.

[6] R. K. Jenamani, P. Sundaresan, M. Sakr, T. Bhattacharjee, and D. Sadigh, "Flair: Feeding via long-horizon acquisition of realistic dishes," *arXiv preprint arXiv:2407.07561*, 2024.

[7] N. Ha, R. Ye, Z. Liu, S. Sinha, and T. Bhattacharjee, "Repeat: A real2sim2real approach for pre-acquisition of soft food items in robot-assisted feeding," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 7048–7055.

[8] E. K. Gordon, X. Meng, T. Bhattacharjee, M. Barnes, and S. S. Srinivasa, "Adaptive robot-assisted feeding: An online learning framework for acquiring previously unseen food items," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 9659–9666.

[9] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu, "Robot learning on the job: Human-in-the-loop autonomy and learning during deployment," *The International Journal of Robotics Research*, p. 02 783 649 241 273 901, 2022.

[10] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *arXiv preprint arXiv:2012.06733*, 2020.

[11] R. A. Knepper, S. Tellex, A. Li, N. Roy, and D. Rus, "Recovering from failure by asking for help," *Autonomous Robots*, vol. 39, pp. 347–362, 2015.

[12] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy, "Asking for help using inverse semantics," 2014.

[13] A. Z. Ren, A. Dixit, A. Bodrova, *et al.*, "Robots that ask for help: Uncertainty alignment for large language model planners," *arXiv preprint arXiv:2307.01928*, 2023.

[14] J. F. Mullen Jr and D. Manocha, "Towards robots that know when they need help: Affordance-based uncertainty for large language model planners," *arXiv preprint arXiv:2403.13198*, 2024.

[15] R. Papallas and M. R. Dogar, "To ask for help or not to ask: A predictive approach to human-in-the-loop motion planning for robot manipulation tasks," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 649–656.

[16] T. Fitzgerald, P. Koppol, P. Callaghan, *et al.*, "Inquire: Interactive querying for user-aware informative reasoning," in *6th Annual Conference on Robot Learning*, 2022.

[17] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012, pp. 17–24.

[18] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*, vol. 52, Elsevier, 1988, pp. 139–183.

[19] J. B. Smith, P. Baskaran, and J. A. Adams, "Decomposed physical workload estimation for human-robot teams," in *2022 IEEE 3rd International Conference on Human-Machine Systems (ICHMS)*, IEEE, 2022, pp. 1–6.

[20] A. Hurst, A. Lerer, A. P. Goucher, *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[21] Z. Zhou, T. Jin, J. Shi, and Q. Li, "Calibrating llm confidence with semantic steering: A multi-prompt aggregation framework," *arXiv preprint arXiv:2503.02863*, 2025.

[22] P. Chhikara, "Mind the confidence gap: Overconfidence, calibration, and distractor effects in large language models," *arXiv preprint arXiv:2502.11028*, 2025.

[23] Y. Huang, J. Song, Z. Wang, *et al.*, "Look before you leap: An exploratory study of uncertainty measurement for large language models," *arXiv preprint arXiv:2307.10236*, 2023.

[24] R. Banerjee, R. K. Jenamani, S. Vasudev, A. Nanavati, S. Dean, and T. Bhattacharjee, "To ask or not to ask: Human-in-the-loop contextual bandits with applications in robot-assisted feeding," *arXiv preprint arXiv:2405.06908*, 2024.

[25] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, pp. 1–13, 2020.

[26] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.