






Leveraging Large Language Models for Preference-Based Sequence Prediction

Michaela Tecson¹^a, Daphne Chen²^b, Michelle Zhao¹^c, Zackory Erickson¹^d
and Reid Simmons¹^e

¹Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A.

²Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, Washington, U.S.A.

Keywords: Meal Preparation, Sequence Prediction, User Preferences, Assistive Technology, Cooking Assistance, Large Language Models, Action Anticipation.

Abstract: We present a novel approach to leveraging Large Language Models (LLMs) for action prediction in meal preparation sequences, with a focus on tailoring predictions based on user preferences. We introduce methods using OpenAI's GPT-4o model to predict subsequent actions in a sequence by providing different forms of context such as sequences from other participants or prior sequences of the test participant. Our approach outperforms baseline methods, including Aggregate Long Short-Term Memory (LSTM) and mixture-of-experts (MoE) models, by up to 33.8% by leveraging the LLM's ability to adapt predictions based on minimal prior context. We highlight the generalizability of the method across different cooking domains by analyzing the results on two different cooking datasets. This adaptability will be useful for assistive systems aiming to support older adults, especially those with Mild Cognitive Impairments (MCI), in completing complex, sequential tasks in ways that align with the user's preferences. The full prompts used in this work can be found at the project webpage: sites.google.com/view/preference-based-prediction.


1 INTRODUCTION


Older adults, particularly those with Mild Cognitive Impairments (MCI), often encounter significant challenges when performing complex, sequential tasks such as meal preparation (Tuokko et al., 2005; Wherton and Monk, 2010). Those with MCI may struggle with maintaining their independence in their daily routines, which can drastically undermine their sense of well-being (Yu et al., 2023). Difficulties in cooking caused by cognitive decline can also severely impact their ability to meet nutritional needs, which is crucial for maintaining overall health. The nutritional intake and dietary habits of individuals with MCI are especially important to maintain (McGrattan et al., 2018), yet meal preparation is an instrumental activity of daily living (IADL) that proves to be a challenge faced by many older adults with MCI everyday (Johansson et al., 2015).


Assistance that takes into account the current needs and long-standing habits of these individuals is more likely to be effective and adopted by the target population (Sanders and Martin-Hammond, 2019). Additionally, prior work has shown that in-situ feedback systems significantly improve performance in individual tasks for those with cognitive impairments (Funk et al., 2015). Therefore, tailoring assistance that acts in line with their personal preferences will improve user acceptance, enhancing their quality of life while maintaining their independence in daily activities (Randers and Mattiasson, 2004).


In this work, we present a novel approach to preference-based sequence prediction using large language models (LLMs). While it may appear that there are limited preferences to habitual meal preparation tasks such as preparing a salad or a sandwich, our work uncovers that there are multiple different preferences for even simple meals, further motivating the need for preference-based assistance.


We present two unique approaches to preference-based sequential reasoning that each leverage the capabilities of LLMs: **Independent Context** and **Shared Context**. Our methods are able to dynam-

^a <https://orcid.org/0009-0004-0611-0400>

^b <https://orcid.org/0009-0003-0382-7652>

^c <https://orcid.org/0000-0001-7522-2300>

^d <https://orcid.org/0000-0002-6760-6213>

^e <https://orcid.org/0000-0003-3153-0453>

ically adapt their predictions based on observations of both the user’s prior sequences and the current context of actions that are taking place. Using prior sequences to infer preferences, we can more accurately predict what action a user will take next in a cooking sequence. We show that both of these LLM methods outperform baseline methods for sequence prediction and prior work on preference-based prediction. This approach could assist users, providing guidance when they make a mistake or are unsure of the next step, while still respecting their autonomy and long-established cooking routines. The key contributions of this work are:

- We introduce a novel approach of using LLMs for preference-based sequence prediction.
- We evaluate our approach across two meal preparation datasets for multi-sequence preference prediction and assess generalizability.
- We compare and contrast to prior state-of-the-art methods for preference-based sequence prediction, in which we observe a significant improvement over prior work.

2 RELATED WORK

2.1 Learning Preferences for Sequential Tasks

Prior work has used unsupervised clustering to extract human preferences from sequential tasks and then applies those clusters to training specialized models. For example, when exploring the problem of robot adaptation for collaborating with humans in sequential tasks, prior work learns from human teams and adapts its policies based on the preferences it learns from the latent representation (Nikolaidis et al., 2015; Zhao et al., 2022).

Others have studied the effectiveness of using a mixture-of-experts approach to preference-based sequence prediction (Chen, 2023), similar to the unsupervised learning described above. This method uses an autoencoder to obtain the latent representation of sequences and then clusters those representations to reveal different strategies. A predictor is trained on the sequences belonging to each cluster, which serves as the “expert” that can do prediction most accurately on a sequence belonging to that strategy. However, these methods do not generalize well in a real-world implementation because the predictor training set and test set included sequences from the same participant. Additionally, the predictors on both the aggregate LSTM and mixture-of-experts in Chen

et al. (Chen, 2023) cannot incorporate any new strategy information into their models. We show that our method using LLMs can improve on these by adding additional forms of context, even from the test participant, to better inform the prediction.

Other methods explore learning and inferring human preferences in task sequencing using a two-stage clustering approach (Nemlekar et al., 2021). This method effectively clusters user preferences at both the action level and event level to enhance prediction accuracy in assembly tasks. It is applicable to the scenario where a new person executes the same task that the current model has demonstrations for. In contrast, our approach leverages the LLM’s ability to infer the preferences from an unseen participant and quickly adapt its prediction for future sequences, rather than having to rely on existing demonstrations for that particular task.

2.2 Using LLMs for Robotics

Several advancements have taken place in the applications of LLMs for zero-shot and few-shot reasoning for robot task planning problems (Singh et al., 2022; Arenas et al., 2023; Hu et al., 2024; Lin et al., 2023; Liang et al., 2023; Padmanabha et al., 2024). Other works have leveraged the generalization capabilities of LLMs to summarize the task specification to be aligned with a person’s preferences (Wu et al., 2023; Wang et al., 2023). TidyBot (Wu et al., 2023) for example, is bringing tidying preferences to robot grounded actions with just a few training examples. The summarization is then used to create a set of rules for that particular person, which dictates where the items should go. The robot can then execute that set of rules and carry out the tidying task according to the preferences of the user.

Similarly, Demo2Code (Wang et al., 2023) uses demonstrations to summarize the task specification according to the user’s preference and generates personalized code to carry it out. These demonstrations come in the form of detailed descriptions of the robot state and actions and recursively summarizes these from low-level actions to high-level subtasks. From there, the algorithm expands the high-level subtasks to action-level executable task code. However, works involving code generation from demonstrations all require significant preprocessing of the task description and the demonstrations into a compact form before generating executable code. Additionally, these methods cannot dynamically update the executable code to account for changes in user strategy over time, but rather, they must regenerate the code with those new demonstrations to incorporate those changes. This

can be problematic because a user may have multiple or changing preferences for preparing a meal, and the prior work would not capture that. They also generate code with the intention that the robot execute the entire sequence on its own, which means the task execution code is fixed and lacks the flexibility to adapt during the task, as it is entirely pre-defined rather than continually adjusting to new actions. In contrast, our method continuously considers both prior preferences and immediate context, integrating the most recent actions to reason dynamically and select the next action based on current conditions. We propose a method designed to work *with* the user that extends beyond static rule-based execution as in prior work and can adapt to a new user preference immediately. The method does not just take prior sequences into account, but also prior actions that have just occurred at test time. This is particularly crucial in the use of an assistive cooking system where the sequence of actions can vary significantly based on individual preferences and situational context.

2.3 Assistive Cooking Systems

Many existing cooking assistants aim to provide assistance to the users step-by-step (Chan et al., 2023; Kosch et al., 2019; Sato et al., 2014; Hamada et al., 2005; Bouchard et al., 2020). LLM-enabled conversational assistants can provide contextual instructions during cooking tasks (Chan et al., 2023; Le et al., 2023). However, these methods generate their feedback or instructions based on a set recipe. They assume there is a fixed order of steps that must be completed, not taking into account the different preferences that users may have for preparing their meal. This includes varying ingredients, different ordering of key steps, or specific techniques that a user may want to do. In the work presented here, we show how our methods can take into account the ways the same dish can be prepared so that the assistance can be personalized based on the user’s cooking strategy. This is especially important for the use of this technology in people’s own homes where they’d like to carry out the task in a way that aligns with their longstanding habits.

3 METHODS

This section presents three methods of generating predictions that this paper compares: Our method of context-based LLMs, aggregate LSTM, and mixture-of-experts.



Figure 1: Frame captures from both the overhead camera (left) and the headworn GoPro camera (right) for the salad dataset (top) and sandwich dataset (bottom).

3.1 Datasets Used

All the methods described were tested on two cooking datasets (Chen, 2023). We refer to them as the salad dataset and sandwich dataset throughout this paper. Both datasets consist of two camera views for each sequence: top-down views from a ceiling-mounted camera and egocentric view from a headworn GoPro camera. See Figure 1 for examples of these views. There were the same 17 participants in both datasets. Participants were all healthy adults, ranging from 19-82 years old ($M = 37$ years old, $SD = 22$); 65% Male, 35% Female.

For both datasets, participants were instructed to make the salad or sandwich as if making a meal for themselves in their own home (no recipe to follow) with the ingredients they were given. Each participant completed the task five times, with each attempt referred to as a sequence. Throughout the paper, we will refer to these attempts as “Seq. n ,” where n corresponds to the order in which the task was completed, ranging from 1 to 5.

The datasets were both annotated at a task-level granularity, where each step corresponds to a key stage in the process without delving into the fine-grained sub-actions of object handling or preparation. Some examples from the salad dataset are “*cut tomato*”, “*place tomato into bowl*”, “*add salt*”, and “*mix ingredients*”. Some examples from the sandwich dataset are “*place jam onto bread 1*”, “*spread jam onto bread 1*”, “*put bread together*”, and “*cut off crusts*”.

The full list of actions for both datasets is included in the Appendix.

Salad Dataset

- There are 18 unique actions in the dataset.
- Maximum sequence length was 27 actions.

Users were provided with the following:

- **Ingredients:** lettuce, tomato, cucumber, cheese, oil, vinegar, salt, pepper
- **Utensils:** plate, chopping board, bowl, butter knife, small chef’s knife, large chef’s knife, spoons, forks, whisk

Example sequence from the salad dataset:

```
[start, cut lettuce, place lettuce into bowl, cut tomato, place tomato into bowl, cut cucumber, place cucumber into bowl, cut cucumber, place cucumber into bowl, add salt, add pepper, serve salad onto plate, end]
```

Sandwich Dataset

- There were 18 unique actions in the dataset.
- Maximum sequence length was 24 actions.

Users were provided with the following:

- **Ingredients:** bread, peanut butter, jam
- **Utensils:** plate, chopping board, bowl, butter knife, small chef’s knife, large chef’s knife, spoons, forks, whisk

Example sequence from the sandwich dataset (PB refers to peanut butter):

```
[start, place PB onto bread 1, spread PB onto bread 1, place PB onto bread 1, spread PB onto bread 1, place PB onto bread 2, spread PB onto bread 2, place PB onto bread 2, spread PB onto bread 2, place PB onto bread 2, spread PB onto bread 2, place jam onto bread 1, spread jam onto PB on bread 1, put bread together, serve sandwich onto plate, end]
```

3.2 LLMs for Prediction

In typical machine learning approaches, it is computationally expensive to retrain the prediction models on every new demonstration as it becomes available. To generalize preference-based sequence prediction, we leverage the LLM’s few-shot and zero-shot reasoning capabilities. Specifically, we explore different forms of context by providing the LLM with task-specific training examples, including sequences from other participants as well as previously seen examples from the test set. Prior analysis of the meal preparation datasets (Chen, 2023) showed the majority of participants behave consistently across their multiple attempts. Thus, by gradually introducing the test participant’s own prior sequences, the LLM will be able

to use these prior sequence as context for prediction of unseen data.

The raw dataset annotations use underscores instead of spaces for use in the other methods (“serve salad onto plate” for example is “serve_salad_onto_plate”). Initial experiments used the raw annotations with underscores as input to the LLM. However, the final method included preprocessing steps for reducing the number of input tokens: the underscores were removed, and all actions in a sequence were combined into a single string separated by commas, rather than a list of individual strings. This preprocessing has a negligible effect on accuracy but had around a 40% decrease in input tokens.

We start by giving the LLM context about the goal of the task:

“I am going to ask you to predict the next action in a salad making sequence.”

To replicate how traditional machine learning methods would train the model, we initially provided the sequences for *all* participants, called *context participants*, except for one held-out test participant. We define context participants as any participants that are not the test participants whose sequences are provided to the LLM to build context. Note that in our methods, there are always $N-1$ context participants, but the number can be varied in other applications. Providing all sequences from all the context participants had an unrealistic amount of input tokens at every prediction step, since API calls do not retain context memory as in ChatGPT and other LLM chat interfaces. This would lead to millions of context tokens which is costly, and also not sustainable as new sequences might appear and eventually exceed the context token limit. Additionally, some participants’ behaviors may introduce noise rather than helpful patterns, and such a large number of sequences may introduce causal confusion (de Haan et al., 2019), reducing prediction accuracy. Since LLMs are known to excel at zero-shot and few-shot reasoning tasks (Kojima et al., 2023; Brown et al., 2020), we then explored the impact of providing different and more limited amounts of prior context at each call.

One approach, termed **Shared Context (LLM-SC)** and illustrated in Figure 2, leverages the few-shot reasoning capabilities of LLMs. In this approach, we experimented with providing either one or two randomly selected sequences per context participant to the LLM (called LLM-SC1 or LLM-SC2, respectively). This ensures that the LLM receives a diverse set of examples to generalize the sequential patterns from the dataset. Providing the sequences of other participants may help to uncover patterns such as “*place (vegetable) into bowl*” frequently follow-

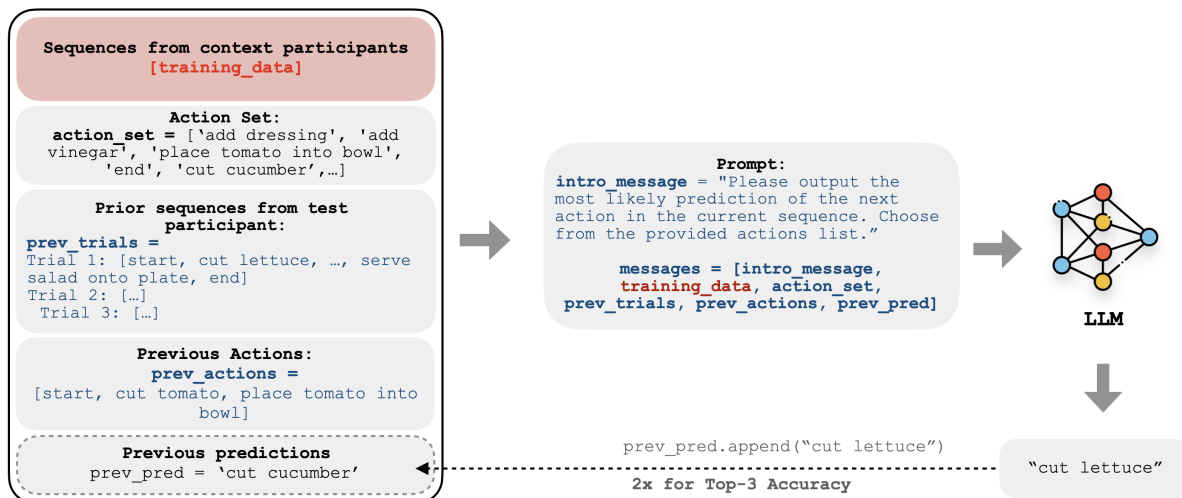


Figure 2: This diagram illustrates all three of the approaches to the LLM methods: Independent Context, Shared Context with one sequence per context participant, and Shared Context with two sequences per context participant. This process shown in the diagram is repeated at every action timestep. The API is called three times per action to obtain the Top-3 accuracy. The previous predictions list is for recording the previous outputs corresponding to first, second, and third most likely predictions. Note that the red box representing the sequences from context participants is only applicable to the Shared Context methods. For a given test participant, the training_data and action_set variables remain fixed across all calls. Additionally, on the first prediction of the first action of the first sequence of the test participant, prev_trials, prev_actions, and prev_pred will all be empty.

ing “cut (vegetable)” or “end” frequently following “serve salad onto plate”. Consequently, this method allows the LLM to better predict unseen sequences without being distracted by excessive or redundant data as in providing all the sequences per context participant.

The alternative approach, referred to as **Independent Context (LLM-IC)**, tests the minimum amount of context the LLM requires to make accurate predictions by using sequences from *only* the test participant as context (see Figure 2). Even if the test participant doesn’t perform the exact same order of actions on each of their five sequences, the general action patterns will still likely hold and be useful for the LLM to reference. For example, if a participant likes to prepare the vegetables, then cheese, then dressing last, the LLM would predict cheese-related actions before dressing actions if it had some prior context about that participant’s preference towards doing that. The rationale behind this approach was that by providing only a minimal amount of context specific to the test participant, the model’s predictions would focus more on the unique patterns of that participant. This would reduce the influence of other participants’ sequences, which could introduce noise.

In both methods, we provided the action bank for the LLM to choose from, since the first test sequence has no other sequences to reference on what actions are allowed. Another key input is the entire list of pre-

vious actions (from a_0 to a_t) that had occurred in the test sequence prior to the one being predicted (a_{t+1}). Earlier iterations of the prompt provided just the single previous action (a_t), but this method showed poor results, due to the lack of full context of what had previously occurred in the sequence.

Our analysis of different prompts showed that the LLM would sometimes output actions outside of the action set, unnecessarily wordy responses, or nonsensical predictions given the context, even with the temperature set to 0. As a result, our final prompts also made specifications about the format of the output:

“Choose from the provided actions list with no other extraneous words or phrases. Your response should be a maximum of W words.”

where W was the number of words in the longest annotation in the dataset.

Lastly, classification methods typically consider a Top- N accuracy when evaluating learning performance (Deshpande and Karypis, 2004; Tang and Wang, 2018). This is particularly relevant for our application to meal preparation, since more than just the top prediction is valuable. In the case of the salad-making task for example, the top prediction after “start, cut tomato, place tomato into bowl” could be “cut cucumber” when the next true action was “cut lettuce”. In this case, “cut cucumber” would have still been a valuable suggestion because it aligns with sequential logic: the prediction continues the veg-

etable preparation phase, the test participant hasn't cut the cucumber yet, and the prediction brings them closer to task completion. Thus, Top-N accuracy captures the variation in possibilities for what the next step might be, which is differentiated by preferences.

We query the API three separate times per action to obtain the top three predictions. The separate calls while providing the most likely, second most likely, etc. returned from previous calls acts as a chain-of-thought process which has been shown to improve performance on reasoning tasks (Wei et al., 2023) and indeed resulted in better accuracy over querying the LLM a single time for the top three most likely prediction all at once.

We iterate over all N participants in the dataset leaving one-subject-held-out. We use the five sequences from the held-out participant as test sequences, testing from Seq. 1 to Seq. 5. For each action (a) within the i -th sequence, the LLM is queried at every timestep t three times to predict the Top-3 most likely predictions of the next action (a_{t+1}) given:

- **Previous Actions.** All previous actions a_0 to a_t
- **Action Bank.** The action bank containing all possible actions for the LLM to choose from.
- **Previously Seen Sequences.** Previously seen sequences of the current test participant (Seqs. 1 to $i-1$).
- **Other Participants' Sequences.** Applies only for Shared Context approaches. 1 or 2 sequences per context participant (excluding the test participant) are randomly chosen. For a dataset with N total participants, this results in either $N-1$ (LLM-SC1) or $2(N-1)$ (LLM-SC2) context participants' sequences, depending on the shared context approach.
- **Previous Predictions.** Applies for the second and third calls to the LLM to predict the second and third most likely predictions to obtain the Top-3 results

Refer to the project webpage¹ for the full prompts used in each of these methods.

In summary, here are the three versions to this approach for using LLMs for action prediction:

- **Independent Context (LLM-IC)** Each participant is treated independently without considering sequences from context participants. The API is called to predict the next action based solely on the actions within the current sequence and if applicable, the test participant's previous sequences.

- **Shared Context (LLM-SC1, 1 Sequence per Context Participant)** One sequence from each of the context participants is included in the context. These sequences are randomly selected per participant, providing the model with additional information about potential action patterns from different sources.
- **Shared Context (LLM-SC2, 2 Sequences per Context Participant)** This approach extends the shared context by including two sequences from each other participant, also randomly selected.

Refer to Figure 2 for an illustration of these LLM methods.

3.3 Baseline Methods

3.3.1 Aggregate LSTM

As a benchmark for the LLM method, we evaluated the performance of an Aggregate Long Short-Term Memory (Agg. LSTM) network. We mapped both datasets' textual annotations to numeric values representing each action and used a sliding-window approach with a window size of 5 actions. For each window, a state vector was created, incremented at the indices corresponding to the actions, normalized, and appended to the feature vector (*moving_window_x*). The next action following the window was recorded as the label (*moving_window_y*). These feature-label pairs were then aggregated into a training set.

To determine the optimal parameters for the LSTM, a grid search was performed on the salad dataset, testing different values of the LSTM hidden size, number of LSTM layers, and learning rate. The values tested for each of these hyperparameters and corresponding accuracy results are shown in Table 1. Based on the results from the grid search, we used the following parameters for the LSTM architecture:

- LSTM hidden size = 16
- Number of LSTM layers = 1
- Four fully-connected layers: Sizes 4×128 , 128×256 , 256×64 , and $64 \times num.classes$ (18 for both of our datasets)
- Activation function = ReLU
- Loss function = Cross-Entropy
- Optimizer = Adam
- Number of epochs = 5000
- Learning rate = 0.001

¹sites.google.com/view/preference-based-prediction

Table 1: Grid Search Results for LSTM Hyperparameters on the Salad Dataset (Exact Accuracy Only).

Hidden Size	Num Layers	LR	Exact Accuracy
4	1	0.0001	49.0%
4	1	0.001	51.4%
4	1	0.01	51.1%
4	2	0.0001	49.5%
4	2	0.001	52.2%
4	2	0.01	51.2%
8	1	0.0001	52.3%
8	1	0.001	56.8%
8	1	0.01	56.5%
8	2	0.0001	54.0%
8	2	0.001	53.2%
8	2	0.01	53.2%
16	1	0.0001	54.5%
16	1	0.001	57.0%
16	1	0.01	55.5%
16	2	0.0001	53.4%
16	2	0.001	55.0%
16	2	0.01	55.1%

3.3.2 Mixture-of-Experts

Autoencoder to Extract Preferences. The mixture-of-experts (MoE) model uses an autoencoder to identify the different strategies in the dataset. To prepare the sequence data for the autoencoder, all sequences were padded with end markers to match the length of the longest sequence, ensuring consistent input sizes. Each sequence was then converted into a tensor of dimensions $1 \times 1 \times (A \times L)$, where A is the number of possible actions and L is the longest sequence in the dataset.

The autoencoder was trained to extract a 2D latent space representation of the sequences using the Mean Squared Error loss function, Adam optimizer, a learning rate of 0.001, and 100 epochs. K-Means

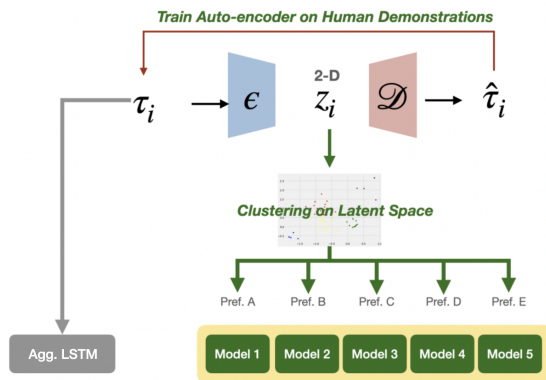


Figure 3: Overview of the Mixture of Experts (MoE) approach compared to the Aggregate LSTM (Chen, 2023). An autoencoder compresses trajectories (τ_i) into a 2D latent space (z_i), followed by clustering to identify distinct strategies. An expert model is trained for each cluster, while the aggregate LSTM is trained on the entire dataset without differentiation.

clustering ($K = 5$) was applied to the latent space, with the number of clusters determined by silhouette score analysis. These clusters reflect distinct strategies participants used to complete tasks. For example, in the salad dataset, one cluster represents sequences where vegetables and dressing are prepared in parallel, while another shows a sequence of preparing vegetables first, then cheese, then dressing. Similarly, in the sandwich dataset, one cluster represents spreading peanut butter on both slices before jelly, while another corresponds to spreading peanut butter on one slice and jelly on the other.

Training the Experts for Prediction. Each sequence in the training data was assigned to a cluster based on its strategy, and five separate LSTMs were trained using the same process and hyperparameters as the aggregate LSTM, with each LSTM focusing only on sequences within its corresponding cluster. This allows each LSTM to specialize in predicting sequences of a specific strategy. To ensure a fair comparison with the aggregate LSTM, which is trained on all sequences except for the test participant, we generated synthetic data to augment the smaller training sets of the expert LSTMs. The synthetic data was created by calculating a probability matrix of action transitions within each cluster and generating sequences based on the most likely actions (Chen, 2023). While these synthetic sequences don't include repeated actions, they still capture the core sequential patterns of each cluster, helping the expert LSTMs to match the data scale of the aggregate LSTM.

To effectively leverage the expert predictors, we first identify which cluster an unknown participant belongs to by maintaining a running belief vector. This belief vector is initialized with equal probabilities across all 5 clusters. For each action, the outputs from all 5 expert predictors are compared to the actual action taken, and the belief vector is updated using a Bayesian update rule.

During evaluation, the expert predictor corresponding to the highest belief is used to predict the next action. If there's a tie, the lowest-numbered strategy is selected. The Bayesian update allows the belief vector to adapt over time, enabling the model to adjust if a participant shifts strategies during the task. This continuous belief update process ensures that the model remains responsive to changing patterns in the participant's actions.

4 RESULTS

The evaluation process involved leave-one-subject-out, testing on the 5 sequences of the held-out subject. The performance of each method was averaged over all test participants. All of the models predict the next action, one action at a time.

Two accuracy metrics are used:

- **Top-1 Accuracy.** The number of times the most likely predicted action matches the actual next action, divided by the total number of actions.
- **Top-3 Accuracy.** The number of times the correct action is within the top three predicted actions, divided by the total number of actions.

We used OpenAI’s GPT-4o model (OpenAI, 2024) for running all of the LLM experiments.

Top-1 Accuracy Results

Table 2: Salad Dataset Top-1 Accuracy of Different Methods Across Sequences.

Model	Seq. 1	Seq. 3	Seq. 5
Agg. LSTM	54.5% ± 0.1%	55.3% ± 1.6%	52.3% ± 2.0%
MoE	56.9% ± 2.6%	53.7% ± 2.5%	54.2% ± 2.3%
LLM-IC	53.9% ± 3.1%	83.6% ± 1.1%	86.1% ± 0.5%
LLM-SC1	64.2% ± 1.3%	76.0% ± 3.3%	79.9% ± 2.8%
LLM-SC2	64.1% ± 1.7%	77.0% ± 0.8%	82.4% ± 2.4%

Table 3: Sandwich Dataset Top-1 Accuracy of Different Methods Across Sequences.

Model	Seq. 1	Seq. 3	Seq. 5
Agg. LSTM	69.1% ± 0.6%	66.0% ± 1.3%	56.8% ± 1.6%
MoE	68.7% ± 3.0%	70.3% ± 3.8%	67.6% ± 1.5%
LLM-IC	51.1% ± 5.1%	81.0% ± 2.9%	82.1% ± 0.7%
LLM-SC1	61.3% ± 2.5%	75.6% ± 1.4%	74.8% ± 0.4%
LLM-SC2	60.6% ± 3.2%	76.4% ± 5.0%	74.8% ± 1.9%

Recall that Seq. n refers to the n -th sequence that the participant performed in the study. The order of sequences is maintained during evaluation, with Seq. 1 tested first and Seq. 5 tested last. Seq. n has context on Seq. 1 to Seq. $(n - 1)$. In Tables 2 and 3, we observe that LLM-IC consistently outperforms other approaches in the later sequences (Seq. 3 and Seq. 5) for both datasets in the Top-1 accuracy metric. In the salad dataset, LLM-IC achieves a Top-1 accuracy of 86.1% on Seq. 5, outperforming the aggregate LSTM by 33.8% and the MoE method by 31.9% (Table 2).

LLM-IC also surpasses other LLM-based methods in later sequences. For instance, in Seq. 5 of the salad dataset (Table 2), LLM-IC achieves 86.1%, which is 6.2% and 3.7% higher than LLM-SC1 and LLM-SC2, respectively. A similar trend is seen in the sandwich dataset (Table 3), where LLM-IC reaches 82.1% in Seq. 5, outperforming both LLM-SC1 and LLM-SC2 by 7.3%. Additionally, LLM-IC outper-

forms the aggregate LSTM by 25.3% and the MoE by 14.5% in Seq. 5 (Table 3), highlighting the strength of this method over the baseline methods in later sequences.

In the early sequences (Seq. 1) however, the LLM Shared Context (LLM-SC1 and LLM-SC2) methods outperform other models in the salad dataset. LLM-SC1 achieves 64.2% and LLM-SC2 achieves 64.1%, both outperforming all other methods in Seq. 1 by 7-10% (Table 2). However, in the sandwich dataset (Table 3), the aggregate LSTM achieves the highest accuracy on Seq. 1 of 69.1%, with MoE following closely at 68.7%. The baseline methods continue with a similar level of accuracy in later sequences but are surpassed by the LLM methods. Also, it is important to point out the strength of the baseline methods in Seq. 1 can be expected, since the baseline methods have been trained on $n - 1$ participants’ training data (80 sequences) rather than the LLM methods which have only been given 0, 16, or 32 prior sequences for LLM-IC, LLM-SC1, and LLM-SC2, respectively by that point.

Top-3 Accuracy Results

Table 4: Salad Dataset Top-3 Accuracy of Different Methods Across Sequences.

Model	Seq. 1	Seq. 3	Seq. 5
Agg. LSTM	75.1% ± 1.5%	75.4% ± 2.7%	74.1% ± 2.2%
MoE	82.1% ± 2.9%	76.8% ± 0.3%	76.6% ± 2.8%
LLM-IC	74.6% ± 0.8%	91.4% ± 0.9%	95.0% ± 0.9%
LLM-SC1	80.2% ± 1.5%	82.3% ± 2.2%	85.9% ± 3.4%
LLM-SC2	80.5% ± 0.4%	83.9% ± 1.2%	88.5% ± 1.6%

Table 5: Sandwich Dataset Top-3 Accuracy of Different Methods Across Sequences.

Model	Seq. 1	Seq. 3	Seq. 5
Agg. LSTM	90.6% ± 1.6%	92.0% ± 1.5%	85.1% ± 0.7%
MoE	91.2% ± 1.8%	90.1% ± 0.9%	86.8% ± 2.0%
LLM-IC	69.1% ± 4.4%	90.5% ± 3.3%	89.0% ± 1.8%
LLM-SC1	75.5% ± 4.6%	86.3% ± 3.4%	85.4% ± 0.6%
LLM-SC2	74.5% ± 1.9%	86.3% ± 2.1%	86.0% ± 0.6%

In the salad dataset (Table 4), the LLM-IC method demonstrates strong performance in Seq. 3 and 5, achieving Top-3 accuracies of 91.4% and 95.0%, respectively. Notably, the gap between Top-1 (Table 2, 3) and Top-3 accuracy (Table 4, 5) for LLM-IC is less than 10% in Seq. 3 and Seq. 5 for both datasets, indicating that the correct action is often ranked as the most likely prediction. LLM-SC1 and LLM-SC2 also show smaller gaps between Top-1 and Top-3 accuracy in the salad dataset, with differences of around 6% in Seq. 3 and 5 (Table 2, 4), further supporting the precision of LLM-based predictions. In contrast, the baseline methods exhibit larger discrepancies, with differences between Top-1 and Top-3 accuracy ranging from 20-25%.

Moreover, in the sandwich dataset, the MoE method starts with the highest Top-3 accuracy in Seq. 1 at 91.2%, but its performance declines in later sequences, dropping to 86.8% by Seq. 5. In contrast, LLM-IC begins with a lower Top-3 accuracy of 69.1% in Seq. 1 but surpasses MoE by Seq. 3, reaching 90.5%, and maintains this advantage in Seq. 5. This declining accuracy between Seq. 3 and 5 occurs for all methods in the sandwich dataset (Table 5). However the decrease in accuracy of the baseline methods between Seq. 3 and Seq.5 is 6.9% and 3.3% for aggregate LSTM and MoE, respectively, while the difference is only 1.5% for LLM-IC, 0.9% for LLM-SC1 and 0.3% for LLM-SC2.

5 DISCUSSION

Accuracy on Later Sequences. The LLM Independent Context (LLM-IC) method demonstrated up to 33.8% greater accuracy than other methods when testing on Seq. 5. This notable improvement highlights the LLM’s ability to quickly adapt its prediction model to individual participant preferences with minimal data. The model excels in environments where participants exhibit consistent behavior patterns, even in cases involving outlier participants. The LLM-IC method performs well on later sequences compared to Seq. 1 because it only uses the current test participant’s previous sequences for context.

The strength of the LLM methods can be observed particularly in the jump in accuracy from Seq. 1 to Seq. 3. For example, in the sandwich dataset, the baseline methods outperformed the LLM methods on Seq. 1 (Table 3). However, on Seq. 3, the LLM methods surpass baseline methods by 5.3%-15.0%. The improved performance of the LLM methods over the aggregate LSTM and MoE methods can be attributed to the fact that the latter two methods cannot change their prediction model at test time, whereas the LLM method can use the test participant’s prior sequences to inform the future predictions. While this isn’t the same as updating the model parameters, it is able to incorporate new information on the test participant’s strategy, while the aggregate LSTM and MoE methods can only rely on the strategies that already existed in the training set. This can be seen in the results from both datasets, where both the aggregate LSTM and MoE methods have a static or even downward trend between Seq. 1 to Seq. 5, while all three of the LLM methods have an overall upward trend from Seq. 1.

In Seq. 3 and Seq. 5 for both datasets (Table 2 and 3), the LLM-IC method consistently outperforms all other approaches in Top-1 accuracy, includ-

ing the other LLM-based methods. This superior performance is likely due to LLM-IC’s ability to leverage the increasing amount of information about the test participant in later sequences, which allows it to make more informed predictions. In contrast, although the LLM Shared Context (LLM-SC) methods also gain access to more of the test participant’s sequences over time, the addition of sequences from other participants acts as noise. This noise makes the predictions less focused on the test participant’s specific preferences, reducing accuracy.

In the sandwich dataset, there was a noticeable decrease in Top-3 accuracy across all methods between Seq. 3 and Seq. 5 (Table 5), indicating that participants began to deviate from the strategies they followed in earlier trials. Despite this decrease, providing context from prior trials consistently outperformed Seq. 1 results for all of the LLM methods, but not for the baseline methods. Notably, LLM-IC demonstrated significantly better performance compared to other methods on Seq. 5 (Table 3). While baseline methods showed a consistent downward trend in accuracy from Seq. 1 to Seq. 5 in both datasets (Table 4, 5)—likely due to their inability to adapt to participants’ evolving strategies—the LLM methods exhibited an upward trend, highlighting their superior adaptability to changing patterns.

Performance on Unseen Data. When testing on completely unseen data (Seq. 1), the LLM Shared Context (LLM-SC1 and LLM-SC2) methods outperformed all other methods in Top-1 accuracy in the salad dataset. This indicates that the salad making task has more variations in strategy which the aggregate LSTM cannot capture with the single model, but existing salad-making priors in the LLM can distinguish those.

On the other hand, it seems that the aggregate LSTM and MoE were able to capture the patterns for the sandwich task quite effectively (Table 3, 5), likely because making a peanut butter and jelly sandwich has fewer possible strategies and a smaller action space. The better performance of the baseline methods over the LLM methods in the sandwich dataset on Seq. 1 (Table 3), indicates that these methods could be suitable for simpler meal preparation tasks with a small action space, limited number of possible strategies, and when there’s little prior information available about the participants’ previous sequences. Thus, our LLM methods are ideal candidates for general use such as assisting with a variety of recipes, more complicated meal preparation tasks involving more variation in preferences, or when there is prior information available about the user’s previous preferences.

By directly leveraging the sequences from other participants in LLM-SC1 and LLM-SC2, the model is able to more effectively learn the general patterns as well strategies that exist for the task. The shared context allows the LLM to generalize better to new participants and unseen data. We can see this by comparing the LLM-SC1 and LLM-SC2 average accuracy on Seq. 1 with that of the LLM-IC. LLM-SC1 and LLM-SC2 outperforms LLM-IC significantly on Seq. 1 for both datasets. Thus, the context provided by other participants proves to be useful when there are no prior sequences available from the current test participant. The tradeoff in prediction accuracy between the LLM-IC method and LLM-SC methods over the trials shows that there may be some room for overall improvement by leveraging a mix of both strategies: providing sequences from other participants when prior sequences of the test participant are unavailable, then removing the other participant sequences as context once the test participant’s prior sequence can be referenced.

Overall, the LLM methods exhibit significant improvements in prediction accuracy over the baseline methods, as shown in the Seq. 3 and Seq. 5 accuracies. LLM-IC shows adaptation to personal preferences with minimal data, while LLM-SC excels in predicting actions in completely unseen data. These findings underscore the effectiveness of LLMs in enhancing action prediction through prior information about their preferences towards a certain strategy.

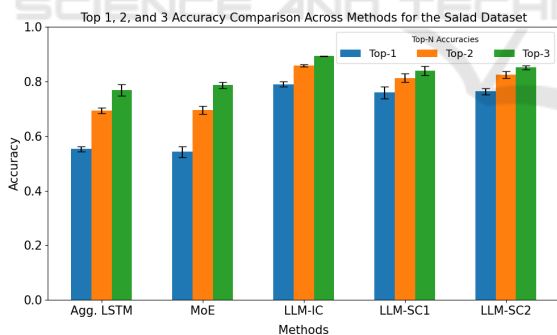


Figure 4: Comparison of average overall Top-1 to Top-3 accuracy for all five different methods for the Salad Dataset.

Top-N Accuracy Comparison. Figures 4 and 5 compares the Top-N results. For both datasets, the LLM-based methods (LLM-IC, LLM-SC1, LLM-SC2) show a smaller difference between the Top-1, Top-2, and Top-3 accuracy compared to the baseline methods (aggregate LSTM, MoE). In particular, the accuracy gains from Top-1 to Top-2 and Top-2 to Top-3 are smaller, indicating that the LLM methods are predicting the correct action as the most likely prediction more often than the baseline methods. The base-

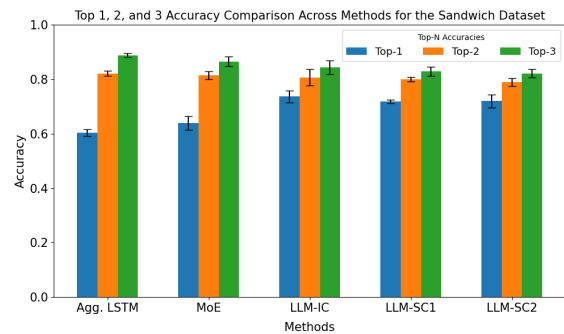


Figure 5: Comparison of average overall Top-1 to Top-3 accuracy for all five different methods for the Sandwich Dataset.

line methods in both dataset have a large gap between Top-1 and Top-2 prediction accuracy and a smaller gap between Top-2 and Top-3, indicating that these are predicting the correct action mostly as the first and second most likely predictions.

This comparison of Top-N accuracies demonstrates the tradeoff between computational efficiency and different accuracy metrics. In the sandwich dataset (Figure 5), the Top-3 accuracies across all methods is similar. Thus, from a deployment standpoint, even the baseline methods can be considered if the Top-3 accuracy is the chosen metric for the specific sandwich-making task. However, the Top-3 results from the salad dataset are less consistent across methods. The LLM methods outperform the baseline methods for all three Top-Ns (Figure 4), indicating that there must be different considerations made for accuracy metrics and methods used in deployment depending on the dataset.

Generalization to Other Cooking Domains. The similarity in overall trends from both the salad dataset and sandwich dataset indicate that the LLM methods can generalize to multiple cooking domains. Making a salad or a peanut butter and jelly sandwich are relatively common meal preparation tasks and are likely described in numerous online recipes, tutorials, and articles. This makes it likely that the LLM has been exposed to a diverse set of examples of ways to make the same dish. As a result, the model has developed robust priors that we can leverage effectively to generalize sequence prediction for a variety of meal preparation tasks with different preferences.

6 FUTURE WORK

One interesting next direction would be an LLM Adaptive Context approach, combining the LLM-IC

method and LLM-SC methods to further improve prediction performance. We could use the LLM-SC method on Seq. 1-2 then switch to LLM-IC on the remaining sequences. By leveraging the ability of the LLM-SC method to perform well on completely unseen data (Seq. 1) and the strength of the LLM-IC on later sequences, we can develop an even more robust prediction system that can quickly adapt to a user's preferences towards making a meal.

Another way to strengthen the performance of LLM methods during Seq. 1 testing (when no prior sequences are available) is to use the aggregate LSTM to generate a reference sequence. This generated sequence can then be provided to the LLM methods as context, serving a similar role to prior participant data in cases where such data is not yet available. The strength of the aggregate LSTM method on the sandwich dataset in particular makes this a promising next step to improve the LLM method performance without prior participant data.

Future work will consist of using these prediction methods that we presented in this paper to contribute to a robust assistive system to help older adults with meal preparation. We have seen that our prediction methods are improved when informed by prior context about a user's preferences, so this will be leveraged to provide meaningful assistance that aligns with how they want to complete the task and how they would like to be assisted. The assistive system will provide feedback in the form of verbal cues to provide useful feedback while still maintaining the user's confidence and autonomy.

7 CONCLUSION

In this work, we presented a novel approach to leveraging Large Language Models (LLMs) for action prediction in cooking sequences, with a focus on tailoring predictions based on user preferences. The primary contributions of this work are as follows: First, we introduced the use of LLMs for personalized action prediction, incorporating OpenAI's GPT-4 model to demonstrate how LLMs can effectively predict the next action in a sequence by using the user's previous actions as context. This method outperformed baseline models, such as aggregate LSTM and mixture-of-experts models, due to the LLMs' ability to adapt with minimal prior context. We also compared various contextual approaches, evaluating three distinct methods—Independent Context (LLM-IC), Shared Context with one sequence per context participant (LLM-SC1), and Shared Context with two sequences per context participant (LLM-SC2). This comparison re-

vealed the strengths of each approach, with the Shared Context methods particularly excelling in predictions for unseen participants. Additionally, we demonstrated that the five methods tested, including the LLM approaches, could be generalized to other cooking domains, such as the salad and sandwich datasets. The LLM methods were especially effective due to the vast amount of cooking-related information available in the training data and the robust reasoning capabilities of the models. Finally, our results showed significant improvement over baseline methods, with LLM methods—especially the Independent Context approach—achieving up to 33.8% higher accuracy in later sequences, highlighting the effectiveness of LLMs in quickly and effectively adapting to individual preferences, even with limited data.

This work shows promising potential for helping to develop assistive technologies for those who struggle with meal preparation. By integrating our approach, such technologies can offer meaningful, personalized assistance. This can empower older adults to regain independence and confidence in this essential daily task, ultimately enhancing both their well-being and quality of life.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2112633. We would like to thank Divya Gupta for her work on annotating the sandwich dataset.

REFERENCES

- Arenas, M. G., Xiao, T., Singh, S., Jain, V., Ren, A. Z., Vuong, Q., Varley, J., Herzog, A., Leal, I., Kirmani, S., Sadigh, D., Sindhwani, V., Rao, K., Liang, J., and Zeng, A. (2023). How to prompt your robot: A promptbook for manipulation skills with code as policies. In *2nd Workshop on Language and Robot Learning: Language as Grounding*.
- Bouchard, B., Bouchard, K., and Bouzouane, A. (2020). A smart cooking device for assisting cognitively impaired users. *Journal of Reliable Intelligent Environments*, 6(2):107–125.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

- Chan, S., Li, J., Yao, B., Mahmood, A., Huang, C.-M., Jimison, H., Mynatt, E. D., and Wang, D. (2023). "mango mango, how to let the lettuce dry without a spinner?": Exploring user perceptions of using an llm-based conversational assistant toward cooking partner.
- Chen, D. (2023). Learning task preferences from real-world data. Master's thesis, Carnegie Mellon University, Pittsburgh, PA.
- de Haan, P., Jayaraman, D., and Levine, S. (2019). Causal confusion in imitation learning.
- Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177.
- Funk, M., Mayer, S., and Schmidt, A. (2015). Using in-situ projection to support cognitively impaired workers at the workplace. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '15, page 185–192, New York, NY, USA. Association for Computing Machinery.
- Hamada, R., Okabe, J., Ide, I., Satoh, S., Sakai, S., and Tanaka, H. (2005). Cooking navi: assistant for daily cooking in kitchen. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, page 371–374, New York, NY, USA. Association for Computing Machinery.
- Hu, Z., Lucchetti, F., Schlesinger, C., Saxena, Y., Freeman, A., Modak, S., Guha, A., and Biswas, J. (2024). Deploying and evaluating llms to program service mobile robots. *IEEE Robotics and Automation Letters*, 9(3):2853–2860.
- Johansson, M. M., Marcusson, J., and Wressle, E. (2015). Cognitive impairment and its consequences in everyday life: experiences of people with mild cognitive impairment or mild dementia and their relatives. *International Psychogeriatrics*, 27(6):949–958.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2023). Large language models are zero-shot reasoners.
- Kosch, T., Wennrich, K., Topp, D., Muntzinger, M., and Schmidt, A. (2019). The digital cooking coach: using visual and auditory in-situ instructions to assist cognitively impaired during cooking. In *Proceedings of the 12th ACM International Conference on Pervasive Technologies Related to Assistive Environments*, PETRA '19, page 156–163, New York, NY, USA. Association for Computing Machinery.
- Le, D. M., Guo, R., Xu, W., and Ritter, A. (2023). Improved instruction ordering in recipe-grounded conversation.
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. (2023). Code as policies: Language model programs for embodied control.
- Lin, K., Agia, C., Migimatsu, T., Pavone, M., and Bohg, J. (2023). Text2motion: from natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365.
- McGrattan, A. M., McEvoy, C. T., McGuinness, B., McKinley, M. C., and Woodside, J. V. (2018). Effect of dietary interventions in mild cognitive impairment: a systematic review. *British Journal of Nutrition*, 120(12):1388–1405.
- Nemlekar, H., Modi, J., Gupta, S. K., and Nikolaidis, S. (2021). Two-stage clustering of human preferences for action prediction in assembly tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3487–3494.
- Nikolaidis, S., Ramakrishnan, R., Gu, K., and Shah, J. (2015). Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, page 189–196, New York, NY, USA. Association for Computing Machinery.
- OpenAI (2024). Gpt-4 technical report.
- Padmanabha, A., Yuan, J., Gupta, J., Karachiwalla, Z., Majidi, C., Admoni, H., and Erickson, Z. (2024). Voicepilot: Harnessing llms as speech interfaces for physically assistive robots. *arXiv preprint arXiv:2404.04066*.
- Randers, I. and Mattiasson, A.-C. (2004). Autonomy and integrity: upholding older adult patients' dignity. *Journal of Advanced Nursing*, 45(1):63–71.
- Sanders, J. and Martin-Hammond, A. (2019). Exploring autonomy in the design of an intelligent health assistant for older adults. In *Companion Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19 Companion, page 95–96, New York, NY, USA. Association for Computing Machinery.
- Sato, A., Watanabe, K., and Rekimoto, J. (2014). Mimi-cook: a cooking assistant system with situated guidance. TEI '14, page 121–124, New York, NY, USA. Association for Computing Machinery.
- Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. (2022). Progprompt: Generating situated robot task plans using large language models.
- Tang, J. and Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 565–573, New York, NY, USA. Association for Computing Machinery.
- Tuokko, H., Morris, C., and Ebert, P. (2005). Mild cognitive impairment and everyday functioning in older adults. *Neurocase*, 11(1):40–47. PMID: 15804923.
- Wang, H., Gonzalez-Pumariega, G., Sharma, Y., and Choudhury, S. (2023). Demo2code: From summarizing demonstrations to synthesizing code via extended chain-of-thought.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models.
- Wherton, J. P. and Monk, A. F. (2010). Problems people with dementia have with kitchen tasks: The challenge for pervasive computing. *Interacting with Computers*, 22(4):253–266. Supportive Interaction: Computer Interventions for Mental Health.
- Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., and Funkhouser, T. (2023). Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*.

Yu, R., Lai, D., Leung, G., Tong, C., Yuen, S., and Woo, J. (2023). A dyadic cooking-based intervention for improving subjective health and well-being of older adults with subjective cognitive decline and their caregivers: A randomized controlled trial. *The Journal of nutrition, health and aging*, 27(10):824–832.

Zhao, M., Simmons, R., and Admoni, H. (2022). Coordination with humans via strategy matching.

APPENDIX

Action Sets

Salad Action Set. [start, cut lettuce, place lettuce into bowl, cut tomato, place tomato into bowl, cut cucumber, place cucumber into bowl, add salt, add pepper, serve salad onto plate, end, cut cheese, place cheese into bowl, add oil, add vinegar, mix ingredients, mix dressing, add dressing]

Sandwich Action Set. [start, place peanut butter onto bread 1, spread peanut butter onto bread 1, place peanut butter onto bread 2, spread peanut butter onto bread 2, place jam onto bread 2, spread jam onto peanut butter on bread 2, put bread together, serve sandwich onto plate, end, place jam onto bread 1, spread jam onto peanut butter on bread 1, spread jam onto bread 2, spread jam onto bread 1, cut sandwich, spread jam next to peanut butter on bread 1, cut off crusts, spread peanut butter onto jam on bread 1]

LLM Prompts

These were the prompts used in the Independent Context and Shared Context LLM methods. Temperature was set to 0.

Independent Context

In this setup, the model is called three times to generate the most likely, second most likely, and third most likely predictions. The output of each call is used to guide the next. The second prediction uses the output of the most likely prediction, and the third prediction uses the outputs of both the most and second most likely predictions.

Asking for Predictions with Previously Seen Trials.

“I am going to ask you to predict the next action in a salad-making sequence. For an unseen test participant, I will provide some of their previous trials for reference, as well as the previously seen actions from the current trial. Please output the most likely prediction of the next action which would immediately follow the last action in the provided list of previous actions in the current sequence. Choose from the provided actions list with no other extraneous words or phrases. Your response should be a maximum of 4 words.”

Asking for Predictions Without Previously Seen Trials (Testing on Seq. 1).

“I am going to ask you to predict the next action in a salad-making sequence. You have not seen any of this participant’s previous trials. I will provide the previously seen actions from the current trial. Please output the most likely prediction of the next action which would immediately follow the last action in the provided list of previous actions in the current sequence. Choose from the provided actions list with no other extraneous words or phrases. Your response should be a maximum of 4 words.”

For the second and third most likely predictions, follow the same process as described above:

- Second prediction: *“If the most likely prediction is pred_list, please output the second most likely prediction.”*
- Third prediction: *“If the most likely prediction is pred_list[0] and the second most likely is pred_list[1], please output the third most likely prediction.”*

Shared Context

In this case, the model is again called three times (most likely, second most likely, and third most likely predictions), using the output from the previous call to inform subsequent predictions. Additionally, the model is provided with context from other participants’ trials.

Asking for Predictions with Previously Seen Trials.

“I am going to ask you to predict the next action in a salad-making sequence. For an unseen test participant, I will provide some of their previous trials, as well as the entire trials of other participants for reference. I will also provide the previously seen actions from the current trial. Please output the most likely prediction of the next action which would immediately follow the last action in the provided list of previous actions in the current sequence.”

Choose from the provided actions list with no other extraneous words or phrases. Your response should be a maximum of 4 words.”

Asking for Predictions Without Previously Seen Trials. *“I am going to ask you to predict the next action in a salad-making sequence. You have not seen any of this participant’s previous trials. I will provide the previously seen actions from the current trial as well as the entire trials of other participants for reference. Please output the most likely prediction of the next action which would immediately follow the last action in the provided list of previous actions in the current sequence. Choose from the provided actions list with no other extraneous words or phrases. Your response should be a maximum of 4 words.”*

For the second and third most likely predictions, follow the same process as described in the Independent Context:

- Second prediction: *“If the most likely prediction is `pred_list`, please output the second most likely prediction.”*
- Third prediction: *“If the most likely prediction is `pred_list[0]` and the second most likely is `pred_list[1]`, please output the third most likely prediction.”*

Project Webpage

The prompts used in this work are also available on the project webpage: sites.google.com/view/preference-based-prediction