Habit Formation for Household Service Robots - A Methodology

Kyum Lee, University of Toronto

May 1, 2025

1 Introduction and Motivation

1.1 Personal Robotics > General Robotics

The next step for household service robots is the ability to internalize the user's unique routines and predict when and where tasks will be asked. Additionally, the robot must hold understanding of external events such as calendar events and weather events to further aid in the prediction of tasks that the user will request.

This is a very interesting and under-researched technical problem, which requires the robot to 1) understand the semantics of tasks, 2) remember recurring patterns, and finally 3) build permanent habits that mirror their users.

I propose a novel cognitive architecture combining multi-modal graph neural networks, LLM semantic embeddings, and continual learning practices that not only do everything described above, but can also generalize to all types of tasks and habits (i.e., not limited by task primitives).

2 Overview of Architecture

The proposed architecture has three main components that allow it to learn habits from multi-modal data. We consider the following factors when learning habits:

- **Home state**: e.g., Messiness level of the kitchen, positions of relevant household objects, messiness of the floor, etc.
- **Time state**: Day of the week, time of task request, etc.
- **External state**: Calendar events of the day, weather, etc.

Each of these require unique embeddings to be recognized by the GNN. Therefore, we utilize different pipelines for each of these factors to extract relevant informations from each state, as can be seen below.

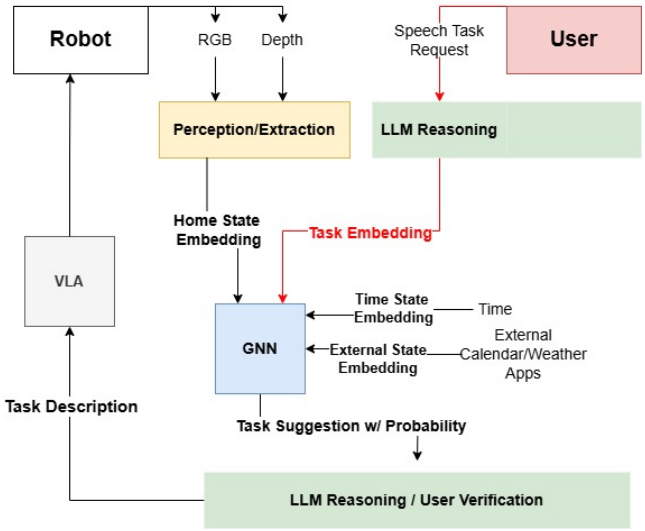

Figure 1. Note that the red pathway is only active during training step.

The rest of this report will deal with each of these blocks and discuss the current planned methodology for implementation, as well as justification of their inclusion.

3 Task Understanding through LLMs

Each episode of data collection for GNN training will begin with the user requesting a task. The robot will also have access to a semantic map of the entire home environment, which will be created at the very beginning through semantic SLAM procedures.

Tasks requested by the user are then encoded as nodes within the memory graph and must capture the semantic nature of the instruction. To achieve robust task understanding, I will embed these commands using a pre-trained language model, such as Sentence-BERT or `text-embedding-ada-002`, producing dense semantic representations that preserve task similarity in the embedding space.

Each task description t_i is transformed into a vector $x_i^{\text{task}} \in \mathbb{R}^d$ via:

$$x_i^{\text{task}} = \text{LLM\_Embed}(t_i)$$

During inference, the GNN will receive these embeddings as node features, allowing the architecture to relate current and past tasks in a semantically grounded way.

Finally, each task will be added to a *task list*, where each embedding now becomes an allowable and possible future action for the robot to perform. This approach allows the robot to maintain safety by not hallucinating brand new actions, and also allows for generality, since the user's past requests are what builds up the allowance list.

4 Perception: Embedding Home State Snapshots

When a task is requested, the robot will move to the site of the task (provided by the LLM embedding), and before any work is done, scan the entirety of the area using its camera. Understanding this captured household context requires embedding the physical state of the environment into a compact but relevant representation.

To quantify home conditions like "messiness," I will deploy computer vision pipelines that extract features such as:

- Number of relevant objects (via YOLOv8)
- Floor coverage ratio (from depth segmentation)
- Surface clutter score (entropy-based texture analysis)
- Ambient metrics (light level, noise, temperature)

These features will then be aggregated into a vector $x_j^{\text{state}} \in \mathbb{R}^k$, which is the embedding for the *home state node*. Normalization of this vector is still something that I am figuring out. This approach allows the graph neural network to reason over states of the home as first-class entities, correlated with task behavior over time.

5 Heterogeneous Graph Neural Networks as a Habit Learning Agent

5.1 HGNNs and Multimodality

Heterogeneous Graph Neural Networks (HGNNs) extend traditional GNNs to operate on graphs with multiple types of nodes and

edges. HGNNs account for this heterogeneity by learning **type-specific message-passing functions** and using relational aggregators that distinguish between interaction types.

In my proposed architecture, HGNNs are crucial for integrating multi-modal data streams into a unified memory graph. Specifically, we model nodes for *task requests* (text-based, semantically embedded) and *home states* (numerical perceptual features), each with distinct feature spaces and temporal behaviors. By leveraging models such as Relational GCN or Heterogeneous Graph Transformer (HGT), we allow the network to learn interaction patterns between semantically dissimilar node types, enabling it to predict user habits and future task requests based on both symbolic and perceptual context.

5.2 Training Through Sub-graphs

As a high-level overview, the robot trains by taking each task node, and taking all other nodes connected by 2 degree separations, then using that sub graph as 1 episode for training.

Formally, if we let $G = (V, E)$ denote the heterogeneous memory graph containing task, home-state, time, and external nodes, then for every task request $v_i^{\text{task}} \in V$ we extract a two-hop induced sub-graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, where $\mathcal{V}_i = \{v \in V \mid \text{dist}(v, v_i^{\text{task}}) \leq 2\}$. The set $\mathcal{D} = \{(\mathcal{G}_i, y_i)\}_{i=1}^{N}$ constitutes the training dataset, with label y_i pointing to the *true* future-task node among the allowed task list $\mathcal{T}$ defined in sections 3 and 4.

The HGNN is trained to minimize a classification loss:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \text{CE}(\text{HGNN}(G_i), y_i)$$

where CE is the cross-entropy loss between the predicted logits and the true task node. This approach ensures that the model learns to associate environmental and temporal contexts with the most likely task, enabling anticipatory behavior. Over time, as the robot collects more episodes, the model incrementally improves its predictive accuracy through accumulated subgraph supervision.

5.3 Reinforcment Learning

Each time a task is suggested by the GNN, I will run a verification step through the user, who will simply approve or deny the suggestion. This does two things: first, it ensures safety by making sure the robot doesn't do anything that the user doesn't expect or doesn't want. Secondly, it can augment the training process by helping the model understand better what the user wants through a reinforcement learning step. This is also something that I am still figuring out how to incorporate into the general training process.

6 Habits as a Continual Learning Problem

We must also approach this problem from a continual learning perspective. This can be done naively by converting every new task request into a local two-hop subgraph that flows into the existing HGNN, then retraining during off-peak hours. However, this becomes unfeasible as the user lives long-term with the robot, accumulating a very large amount of data over years. Instead, we can train on mini-batches that mix these fresh subgraphs with a compact rehearsal buffer of past exemplars, while regularisation (Elastic-Weight Consolidation) and logit-level knowledge distillation protect parameters and decision boundaries vital to earlier habits. Plasticity is confined to lightweight adapter modules inserted in each HGNN layer, so new behaviours can be learned without overwriting core representations.

Let $\{\mathcal{G}_t\}_{t=1}^{\infty}$ be the time-ordered stream of two-hop sub-graphs arriving as the user performs new tasks. I can then update the HGNN online in mini-batches $\mathcal{B}_k$ drawn from two sources:

1. **Recent tasks**: the latest sub-graphs $\mathcal{G}_t$ collected since the previous update. (e.g., over the last 24h)
2. **Rehearsal buffer** $\mathcal{M}$: a bounded memory of *exemplar* sub-graphs selected by reservoir sampling (Chaudhry, 2019). After processing each $\mathcal{G}_t$, we insert it into $\mathcal{M}$ with probability $|\mathcal{M}|/(\text{seen} + 1)$, ensuring an unbiased snapshot of the entire history.

Periodic consolidation. During low-demand hours the robot performs *offline consolidation*: it re-trains on $\mathcal{M} \cup \mathcal{R}$, where $\mathcal{R}$ are sub-graphs *generated* by the HGNN itself (contrastive pseudo-replay). This boosts robustness without increasing on-board memory.

7 Task Expert Through VLAs

Thanks to rapid advances in open-source robot VLAs and datasets, we can perform the suggested task by the GNN through the robot using the VLA as a task expert. We feed in the semantic task description, move to the location using semantic mapping, then run the actions using the VLA.