Interactive Planning-based Cognitive Assistance on the Edge

Zhiming Hu, Maayan Shvo, Allan Jepson and Iqbal Mohomed
Samsung AI Centre, Toronto
What is cognitive assistance?

- One of the most exciting applications in AR Glasses
  - Google Glass, HoloLens 2
- Helpful in a myriad of tasks
  - Health care education and training
  - Industrial tool for remote support
  - Cooking assistant and fitness coach

Image source for HoloLens 2: https://commons.wikimedia.org/wiki/File:HoloLens_2.jpeg, https://creativecommons.org/licenses/by/2.0/legalcode. Changes are not made on the image.
How to build a cognitive assistant?

- Lots of existing work on building cognitive assistance [1,2,3,4]

- Perception module
  - Determine the current task state

- Cognitive module
  - Generate the next step

The motivation

- While it is simple to build a state machine to guide a user to complete some tasks, there are several issues

  - The state machine needs to be pre-defined

  - It cannot list all the possible user errors, thus cannot recover from such failure cases.
How about a planner?

- **Benefits**
  - Flexible, can recover from any user errors

- **Challenges**
  - Need to calculate accurate current task state (CTS)
  - Not as computationally efficient as state machines.
A planning problem

- A planning problem may be encoded in PDDL by defining the domain, initial state, and goal state.
  - \( \text{stack}(x,y) \in A \)
    - \( \text{Pre}_{\text{stack}} = \{ \text{clear}(x), \text{clear}(y), \text{onTable}(x) \} \)
    - \( \text{eff}^+_{\text{stack}} = \{ \text{on}(x,y) \} \) (note: \( x \) is on \( y \))
    - \( \text{eff}^-_{\text{stack}} = \{ \text{clear}(y) \} \)
  - \( G = \{ \text{onTable}(\text{bread1}), \text{on}(\text{ham}, \text{bread1}), \text{on}(\text{lettuce}, \text{ham}), \text{on}(\text{bread2}, \text{lettuce}), \text{on}(\text{tomato}, \text{bread2}), \text{on}(\text{bread3}, \text{tomato}) \} \)

- If all of the ingredients are clear and on the table, one possible solution is \( \pi = \text{stack}(\text{ham}, \text{bread1}), \text{stack}(\text{lettuce}, \text{ham}), \text{stack}(\text{bread2}, \text{lettuce}), \text{stack}(\text{tomato}, \text{bread2}), \text{stack}(\text{bread3}, \text{tomato}) \).

The key to get the correct plan is to obtain accurate current task state.
Ambiguity Resolving

- We keep track of the current task state by recognizing the actions taken since the beginning of the interaction.

- However, we may encounter ambiguous cases where we cannot determine which action was performed by the user.

**Classifier for the Top Object on the Sandwich**

*Sequence of Classification Results:*

- Bread -> Ham -> Bread

- Stack Bread on Ham
- OR
- Unstack Ham from Bread
Dynamic State Tracking

- A planner with state machines
- The planner will only be called when an unexpected action is detected

Figure 3: State tracking with a planner and state machines. The green box shows the current expected action.
We built our system on top of the gabriel-sandwich project with a edge server architecture and adopted the WebSocket to provide the next instruction directly based on the state. The green box shows the current expected action.

To fine-tuning the network, classes in our case, which are bread, tomato, lettuce, ham and could be deployed on a lot of edge devices. We have five on a lightweight MobileNet v2 backbone network adopt the transfer learning method and build our model based on the new list of instructions. The approach of integrating the recognition module and the interactive state tracking module into the planner with a state machine is advantageous since the system will not trigger the planner every time as long as the user is following the plan. Note that we employ here a simple approach to plan execution and monitoring; the rich body of related work offers a myriad of solutions (e.g.,...)

In the edge server, for the image classification module, we place the lettuce instead. An illustrative figure is shown in Figure 3:

![Graphs showing CDF for runtime](image)

(a) Runtime for the planner. (b) Runtime for the classifier

Figure 4: Runtime for the planner and the classifier.

It is feasible to run both the planner and classifier on the edge.
Demo

- The video for our demo is available here.
Future Work

- Personalized instructions
- Resource management for multiple cognitive assistance agents
- Applications that only need partial order
  - Linear Temporal Logic (LTL)
Summary

- We have proposed an architecture for cognitive assistants on the edge.
- Ambiguous task states are prevalent and we need to deal with them.
- We should combine the planner with state machines to enjoy both of the benefits.
Thanks!
zhiming.hu@samsung.com