Agent Verification Layer

Supporting on-the-fly verification of web agent's decisions for screen reader users

The verification layer runs while a browser agent works on a task. At each step, it finds what the user would want to verify (e.g., did the price go up at checkout?) and decides whether the agent should stop and ask the user, simply tell the user as it runs, or log it for user checking at completion.

Scan for a screen reader version

1. Why verify agents?

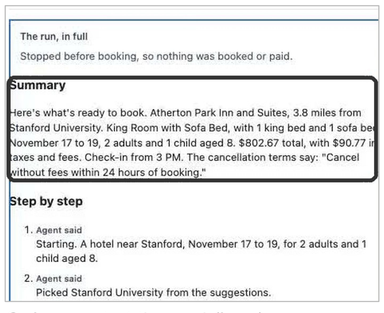
Web agents can now complete real tasks on real websites [1].
We can delegate agents to navigate pages, fill forms, and complete purchases, including on previously **inaccessible** sites.

However, since we are **delegating** tasks, they need to be able to **verify** agents' results against our goals.
When we hand a task to an agent, the agent makes **many small decisions** for you, many of which are not transparent.

When and how should the agent ask the user?

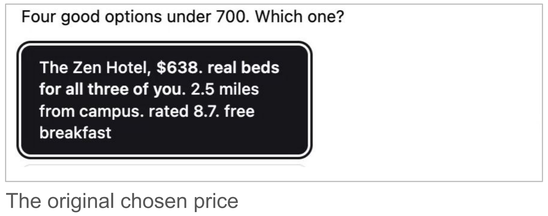


Ask about everything

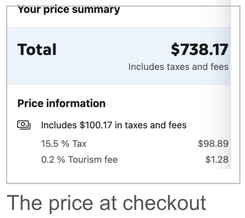


Only report at the end (logs)

- **C1.** If the agent asks about everything, the user is interrupted constantly, so they stop paying attention [2, 3].
- **C2.** If the agent only reports at the end, many actions are too late to fix (irreversible actions).
- **C3.** Users also cannot check the agent's decisions against the live page, which may have changed when they go back.



The original chosen price

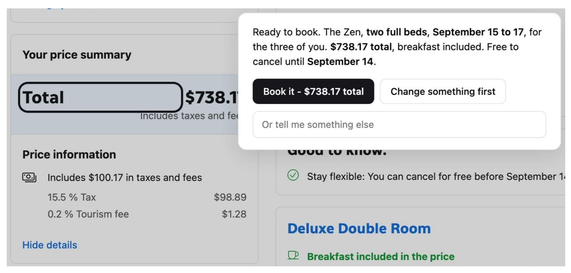


The price at checkout

2. The verification layer

It surfaces decision points to the user **while** the agent is running [4].

- **It needs to be independent of the agent (C3).** Each decision is checked against information on the live page.
- **It needs to check actions before they happen (C2).**

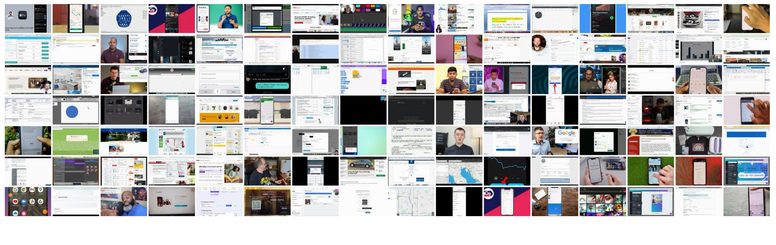


Before booking, the layer reads back the hotel, room, dates, guests and total price.

3. What to ask: Mining a task dataset

How do we know what the decision points people will want to verify for diverse tasks?

We mined ~1.7K videos of people doing the **85 everyday tasks** to find real decision points people have for tasks. The mined dataset includes **~15K decisions points**, organized as a **hierarchical task analysis** (HTA) per task [5].



4. How to ask: Designing a utility function

For each decision point *q*, th verification layer scores each interaction *r* and picks the one with the highest expected utility [5] (addressing C1).

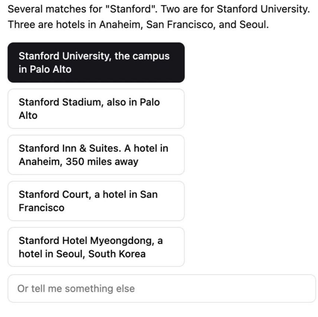
$$EU(r \mid q, t) = P(e \text{ at } n) \cdot P(\text{uncover} \mid q) \cdot D(r \mid q) \cdot I(r) \cdot C_{und}(q) + V_{mon}(q) \cdot A(r) - C_{int}(r, user, t)$$

P(e at n)	how likely something is wrong at this step
P(uncover | q)	how likely this question catches it
C_und(q)	how bad the error is if never caught [3]
C_int(r, user, t)	the cost of interrupting a screen reader user [2]

Three types interactions using an example task: book us a hotel near stanford from sep 15 to 17 for 2 adults and 1 child:

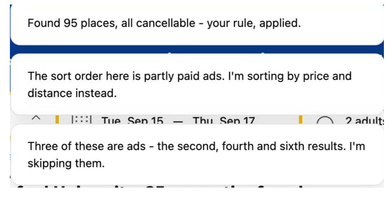
Interactive widget (stop and ask)

"Stanford" can mean the campus, the stadium, or hotels in other cities.



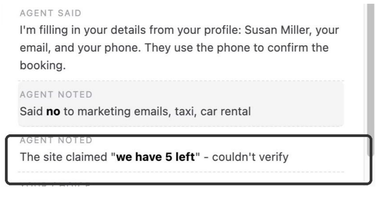
Cognitive checkpoint (inform the user but keep going)

The top results are mostly low-quality paid ads, so it informs the user they it has re-sorted them by price and distance.



Agent log (log it for checking later)

The site said "we have 5 left" for the number of rooms.



[1] Deng et al. 2023. Mind2Web. NeurIPS.
[2] McFarlane 2002. Coordinating the Interruption of People in HCI. HCI.
[3] Parasuraman & Riley 1997. Humans and Automation. Human Factors.
[4] Horvitz 1999. Principles of Mixed-Initiative User Interfaces. CHI.
[5] Annett & Duncan 1967. Task Analysis and Training Design. Occupational Psychology.

Part of the **AI for Accessibility Collective**, with support from Google.org.