

CartTrace-100k: Shopping-Agent Journeys Through Checkout and Payment, and Where They Fail

Pratham Bhatnagar
General Machines

pratham@generalmachines.ai

Deepanshu Pal
General Machines

deepanshu@generalmachines.ai

Abstract

AI agents can now check out and pay through commerce protocols such as UCP, ACP, AP2, MPP and WebMCP, yet no public data shows what happens between a shopper’s request and a paid order. We release CartTrace-100k: 101,416 journeys by one open-weight agent (Qwen3.8-27B) across 1,044 simulated stores in 25 product categories, five protocol surfaces and five payment rails, each against a simulated shopper who reveals some needs only when asked. A code-first judge checks every cart and order against the store’s records and the payment processor’s ledger; 42,102 journeys end in a paid order. The agent succeeds on 68.8% of satisfiable tasks, against 82.2% for a scripted oracle that knows each task. Journeys are lost before payment, at reading product facts and at clarifying needs, and almost never at checkout or payment. Asking is worth the most where needs are hidden (62.1% against 12.4% success with two needs held back), yet the agent asks in 63.3% of journeys where nothing is. Measured within task on checks every protocol can fail, the protocol changes success little and token cost threefold. We release the journeys, the judge’s step-level checks, the oracle’s runs and a kit that replays and re-judges any journey.

Data: hf.co/datasets/generalmachines/CartTrace-100k

Replay kit: github.com/generalmachines/cartTrace-100k

Explorer: machinecommerce.co/carttrace

1 Introduction

Shopping is moving from the web page to the protocol call. The Model Context Protocol gives an agent tools [31]; the Universal Commerce Protocol (UCP) exposes a store’s catalog, cart and checkout [48]; the Agentic Commerce Protocol (ACP) delegates checkout [33]; WebMCP lets a page register its own tools [50]; and AP2 and MPP carry the payment [18, 46]. Platforms, payment networks and standards bodies are building for this because a protocol call is cheaper and more dependable for an agent than a page built for people.

What an agent does once it reaches a store through these protocols has not been measured. Shopping benchmarks stop at a product choice, a cart or a simulated checkout form [14, 27, 35, 44, 49, 52]. Work that does reach a payment protocol studies attacks on one protocol, AP2 [3, 12, 24, 25], or verifies protocols formally [19]. Released agent-payment data covers transfers and API micropayments, not purchases [43, 47]. So the questions a store, a platform or an agent builder would

ask first have no public answer: where between a need and a paid order do journeys fail, does that differ by category and by what the shopper asks for, and does the protocol matter?

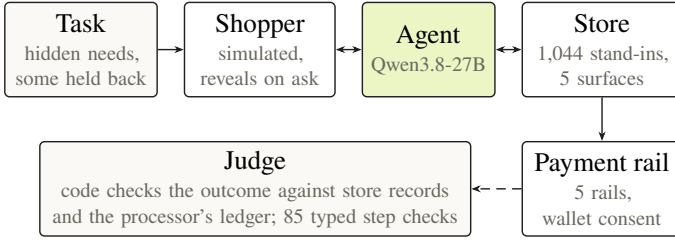
CartTrace-100k answers them on 101,416 journeys. Each runs in a sandbox that serves one of 1,044 stand-in stores, built from open product catalogs in 25 categories and served under one of nine e-commerce platforms’ rules, through one of five protocol surfaces and, when the task allows, one of five payment rails. A simulated shopper holds a hidden specification and keeps some needs back until the agent asks. A judge checks the outcome against the store’s records and the payment processor’s ledger, and answers 85 typed questions about the steps; code settles most of them, and every check records what settled it (Figure 1a). One open-weight agent, Qwen3.8-27B, runs every journey, and a scripted oracle that knows each task runs a tenth of them on the same worlds.

Holding the agent fixed is a choice with a cost. It lets every contrast in the paper (between stores, categories, intents and protocols) be read as a difference in the environment the agent met, and it bought one large, uniform corpus for 971 H100 hours. It also means the levels belong to Qwen3.8-27B: another agent would score differently, and we do not know which contrasts would hold. The replay kit runs any other agent through the same worlds under the same judge (Section 8).

We report six findings (Section 5).

- **Journeys fail before payment.** Most lost journeys fail at reading product facts or at clarifying needs; under one percent fail at building the cart, checking out or paying (Figure 1b).
- **A question is worth something only where a need is hidden.** Within task, asking about a held-back need adds +11.9 points of success; where nothing is held back it adds +1.2, yet the agent asks there in 63.3% of journeys.
- **The agent gives up on tasks the store can serve,** in 7.9% of satisfiable journeys and 27.4% when a substitute would do.
- **What is asked for matters more than the category.** Success varies by 11.0 points across categories and by 36.3 across intents within a category.
- **Protocols change cost more than success.** With the task fixed and on checks every protocol can fail, the clear effects

(a) One journey



(b) Where paying journeys fail

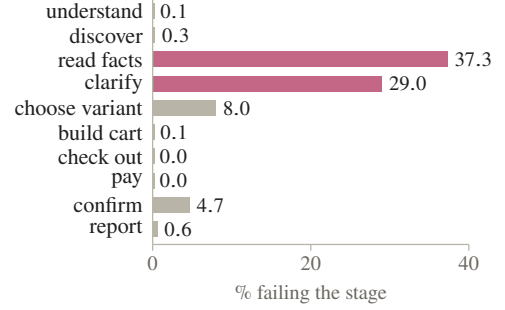


Figure 1. (a) A CartTrace journey. A task fixes what the shopper needs and which needs it keeps back. The agent works the store through one protocol surface and, when allowed, pays through one rail after the shopper approves the total. The judge reads the finished trace and never calls the store. **(b)** The share of Qwen3.8-27B’s paying full-data journeys whose checks at each of the judge’s ten stages score below one half. Journeys are lost at reading product facts and at clarifying the shopper’s needs, not at checkout or payment.

on the odds of success lie between 0.82 and 1.17, while median prompt tokens differ threefold between surfaces at the same success.

- **Weak stores are weak at the first search.** The bottom quarter of stores differ from the top mainly in how often the first search finds a fitting product, and their journeys end in give-ups more than in wrong purchases.

Contributions. (1) CartTrace-100k, the first public set of shopping-agent journeys that check out and pay through several commerce protocols, with the judge’s step-level checks on every journey and an oracle’s journeys on the same tasks. (2) The sandbox, task generator and judge, locked under one specification so that every number here regenerates from the released rows. (3) The findings above, with intervals that re-sample whole stores and protocol comparisons made within task.

2 Related work

Shopping agents. WebShop [52] and τ -bench [5, 53] set the pattern this work follows: a simulated user holds the goal, the agent acts through tools, and the outcome is read from final state. Commerce benchmarks have since widened the catalog [49], moved to live sites [21, 27], spread a task across several shops [35], and hidden the user’s requirements behind clarification [14, 40]. Two results anticipate ours: unverified product attributes are a leading cause of failure [54], and agents do poorly at drawing out what a user wants [39]. Comparing agent interfaces to the same shops, structured interfaces such as MCP beat HTML [44], and a payment provider reports that WebMCP cut tokens and tool calls on its own checkout [45]. None of these runs journeys through checkout and payment on more than one commerce protocol (Table 1).

Agents in markets and on the agent web. Studies of what buying agents choose find position and description effects that sellers can exploit [2, 8], and simulated marketplaces study agents on both sides of a trade [4, 15]. A large live study finds

that sites agents can read are recommended more often [16], and a merchant-side census finds many stores unreadable to declared agents before any checkout [11]; both argue, as Lù et al. [26] do, that the web should be built for agents. We measure the step those studies stop before: the transaction.

Payments. Security work runs agents through AP2’s reference flows to show that a signed cart can still be one the user did not want [3, 12, 24, 25]; formal models cover AP2, ACP, MPP and x402 [19]; and released payment data covers authorisations and micropayments rather than purchases [43, 47]. We run no attacks; our payment faults are designed failures (a declined card, a failed 3-D Secure check), and our question is whether ordinary journeys reach a right, paid order.

Simulated users and judges. Simulated users scale a benchmark but can behave unlike people [13, 32, 36]; simulated customers in particular rarely walk away [6]. Our shopper’s facts are fixed by rules from the hidden specification, and a model only words them. LLM judges carry position, verbosity and self-preference biases [34, 42, 55] and need validating on labelled trajectories [28, 51]. Judges can accept confident verdicts and escalate the rest with guarantees [20]; decision models do this in one pass over a fixed answer set [23], with known sensitivity to wording and negation [38, 41]. We let code settle what state can show and keep model-read checks out of every headline number. We follow the agentic-benchmark checklist [56], cluster intervals by store [30], and document the release as a datasheet [17] in Croissant [1].

3 The environment

Everything runs offline: no journey touches a live store, nothing is bought anywhere, and no card number appears in any trace. A journey has three parts: a sandbox that serves a store and a payment rail, a task with a simulated shopper, and a judge that reads the finished trace (Section 4). Appendix A defines the terms used below.

Table 1. Where agent shopping and payment studies end. *Ends at*: the furthest point a journey reaches. *Categories*: whether results are compared across product categories (–: no such comparison in our reading).

Work	Ends at	Commerce protocols	Categories
WebMall [35]	checkout form	none	–
DeepShop [27]	product choice	none	5
EComAgentBench [14]	product choice	none	–
Agent interfaces [44]	checkout form	MCP, NLWeb	–
Agentic Commerce World [15]	transaction state	own protocol	–
AP2 attacks [24, 25]	signed mandate	AP2	–
CartTrace-100k	paid order	UCP, ACP, WebMCP, AP2, MPP	25

Stores. A stand-in store takes a category, a market and a shape from a real store and serves products from open catalogs: Amazon Berkeley Objects [10], WANDS [7] and the Lazada listings of ShoppingBench [49], sorted into 25 categories. Prices and stock are synthetic, in US dollars. No product, price, reply or identity of a real store is in the release. Real stores do not all implement a protocol the same way, so each stand-in runs under a *platform pack* that reproduces one platform’s rules (search, discount stacking, shipping choice, refused cart lines, MCP handshake): Shopify, WooCommerce, a Magento-like pack, BigCommerce, Medusa, Saleor, Vendure, PrestaShop and a plain dialect.

Protocols and payment. Specification files are vendored and pinned. A journey reaches its store through one of five surfaces: UCP over MCP, UCP over REST, WebMCP (tools registered by the store’s pages in a real Chromium), ACP over REST, or AP2 (intent, cart and payment mandates). A paying journey uses one of five rails: a UCP payment processor, Shop Pay, ACP delegated payment, AP2 mandates, or MPP (an HTTP 402 challenge the wallet answers). The shopper approves the payment where the protocol asks the buyer, and a paying call succeeds only with a token issued after the shopper approved the current total. Every rail settles through a payment-processor simulator. Surfaces and rails form 13 cells: ten that pay and three cart-only.

Tasks and the shopper. Every journey has a hidden specification the agent never sees: predicates over the store’s own product data (a product kind, a brand, an option, a budget, a quantity, a fit, an allergen, a superlative). Each predicate is *stated* in the opening message, *kept* until the agent asks about it, or left to a default. The generator computes which variants satisfy every predicate, so a cart is checked by membership; a task with no satisfying variant is impossible, and refusing is its right end. Tasks come from ten intents (discover, compare, specify, substitute, bundle, gift, replenish, express, budget, policy; Appendix D) and 30 personas that set how much a buyer volunteers, the budget margin and the delivery preference. The shopper is a program: rules decide every fact of a reply, a model words it in the persona’s voice, and code checks the words. It reveals a detail only when asked, never invents one, and consents to pay only the total it was shown.

Full data and designed failures. Most journeys are *full data*: a satisfiable task and a persona built to pay, half stopping at a verified cart and half going on to pay. The rest are *designed failures*, scored separately: impossible tasks, payment faults (a declined card, a failed 3-D Secure check, no usable instrument) and checkout faults forced at completion (a reprice, stock gone, an unshippable address).

Agents. Qwen3.8-27B [37], served by vLLM [22], is the agent under test, with one system prompt for every journey (Appendix E). A scripted oracle that reads the hidden specification runs on a tenth of the tasks, on the same world, task, seed and cell. It bounds what the world and the labels allow; it is not a second model under test.

The run. Stores, packs, prompt and sandbox code sit under one lock whose digest, v1s-9706931d3f3d, every journey carries. The run went from 2026-10-17 to 2026-10-22 in one pass on 8 H100 GPUs (971 GPU hours, 29.9 billion tokens read and written). Journeys hit by a system error were quarantined and rerun under the same id. Table 2 summarises the corpus and Appendix F gives the full accounting.

4 The judge

The judge (judge-1.0) reads a finished journey and never calls the store, so a journey always gets the same verdict and anyone can re-judge it from the released data.

The outcome is settled by code. Code checks the final cart, order or refusal against the hidden specification, predicate by predicate, and classes the journey as right, right by luck (a need the agent never learned happened to hold), wrong on a hidden need, wrong, a correct refusal, a wrong refusal, or no answer. Whether an order was paid, and for how much, is read from the payment processor’s record, never from the agent’s words. The released labels (labels 1.1) then hold every verdict to the store’s own transaction record: a journey rated right whose record disagrees (the cart was never shown to be right, the checkout was left open, the paid order was not the right one) or that carries a safety finding is published as wrong, with the reason. This affects 14,888 journeys. *Success* is right, right by luck or a correct refusal after that correction. *Grounded success* drops right-by-luck and requires every claim the agent made to be backed by what it saw. A gate voids the 8 journeys that the harness, not the store or the

Table 2. The corpus. Success is a right cart or order, or a correct refusal; the interval resamples whole stores (95%). The two agents are never pooled, and designed failures are scored apart from full-data journeys.

	Qwen3.8-27B	Oracle
Journeys released / judged valid	101,416 / 101,408	10,132 / 10,125
full data (stop at cart / go on to pay)	97,432 (48,625 / 48,807)	9,732
designed failures (payment / checkout / impossible)	2,006 / 947 / 1,023	
Stores / categories / tasks / personas	1,044 / 25 / 28,985 / 30	
Surfaces / rails / cells / platform packs	5 / 5 / 13 / 9	
Success, full data (%)	68.8 [68.2, 69.3]	82.2
stop at cart / go on to pay	73.4 / 64.2	
Grounded success, full data (%)	65.7	76.6
Paid orders	42,102	

agent, decided.

The steps are typed questions. Beyond the outcome, the judge asks 85 typed questions per journey in ten stages, from understanding the request to reporting the result (Appendix B). Code answers 71 of them from recorded state. A decision model, Cloudflare’s Clef [9], reads the other 14. Each question has a fixed answer set and a safe default, and sees only an evidence slice holding the parts of the journey it names (the store’s replies for a product fact, the agent’s messages for a claim, the shopper’s words for a need), never the outcome or the answer key. The two stages where journeys fail, reading product facts and clarifying, are settled by code in 99.8% and 72.4% of steps.

What is validated. One model-read question counts towards results: whether the agent’s account of its last payment matches the processor’s record. It passed a gate of 100 human labels at 90% agreement and matched the processor on 100.0% of 41,940 paid attempts. Every other model-read question runs in shadow, recorded and never counted, so no headline number rests on a model’s reading. Clef shares its backbone with the agent, which is a further reason to keep it in shadow until human-label rounds promote each question type.

5 Results

All results are on Qwen3.8-27B’s 97,432 full-data journeys, judged valid, with designed failures scored apart. Intervals resample whole stores. Qwen3.8-27B succeeds on 68.8% [68.2, 69.3] and its grounded success is 65.7%. The oracle succeeds on 82.2% of its full-data journeys under the same labels; on the 11,413 Qwen3.8-27B journeys that share a task, cell and persona with an oracle journey, the oracle succeeds on 82.6% and Qwen3.8-27B on 68.8%. The oracle falls short of 100% where the transaction record cannot certify a right answer (Section 5.7), so 82.2% is the ceiling the labels allow, not evidence of broken tasks.

5.1 Journeys fail before payment

Figure 1b gives, for each of the judge’s ten stages, the share of paying journeys whose checks at that stage score below one half. Two stages carry the loss: reading product facts (37.3%)

and clarifying the shopper’s needs (29.0%). Building the cart, checking out and paying each fail in under one percent, on every protocol. The checks behind the first two are concrete: the agent carts straight from search results without opening a product page in 37.6% of journeys, and hands over the cart without reading it back in 50.7%.

Paying is safe but costly. Qwen3.8-27B placed 42,102 paid orders. Across all designed failures that fired, the designed stop held in 99.7% (1,999 of 2,005); the agent paid when it should not have in 6 of 3,572; its final message reported what happened truthfully in 99.3%; and no impossible task was ever paid. A paying journey takes 54 turns against 23 for one that stops at the cart, and 2.5 times the prompt tokens; 68% of its tokens come after checkout opens.

5.2 A question is worth something only where a need is hidden

The task decides how many needs the shopper keeps back; the agent decides whether to ask. Splitting success by both is the one place the design makes the value of a question visible (Figure 2). With two needs kept back, journeys where the agent asked succeed 62.1% of the time and journeys where it did not, 12.4%. With nothing kept back, asking goes with slightly lower success, yet Qwen3.8-27B asks in 63.3% of those journeys, and asks again for a need the shopper already stated in 40.9% of journeys where that check applies. Raw rates mix tasks, so we also compare runs of the same task with and without a question about the kept-back need: asking adds +11.9 points (+10.2 to +13.7, 2,645 tasks) where a need is kept back and +1.2 points (-0.2 to +2.7) where none is. The fix is targeting, not frequency.

5.3 The agent gives up on tasks the store can serve

7.9% of satisfiable journeys end with the agent refusing, and every such refusal is a give-up; on substitute tasks, where the named product is out of stock and a similar one would do, the rate is 27.4%. On the 1,023 impossible tasks Qwen3.8-27B refuses correctly in 77.4%, so only 9.4% of all its refusals are right. Failure is visible early: a gradient-boosted model given a journey’s first twelve actions predicts failure with AUC 0.73 on stores it has not seen.

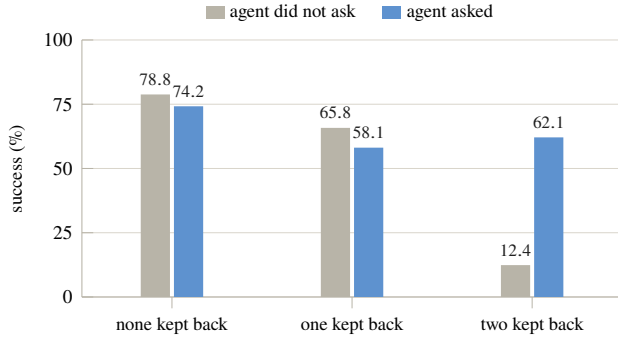


Figure 2. Success by how many needs the shopper keeps until asked, and whether the agent asked.

5.4 What is asked for matters more than the category

Success by category runs from 63.9% (books & stationery) to 74.9% (jewelry & watches), a spread of 11.0 points (Figure 3). Under a store-clustered test with Holm correction, 2 categories sit below the mean category and none above. False refusals vary more than success, by a factor of 3.0 (4.3% to 13.1%), so much of what makes a category hard is the agent giving up in it.

The intent within a category matters more (Figure 4). The average spread across intents inside a category is 36.3 points, against 29.4 across categories inside an intent. Two intents form the hard corner of every category: substitute (52.3%) and discover, a need without a product name (55.6%), against 69.6% for express. Of the 18 category–intent cells below 50%, 10 are substitute and 6 discover; the lowest is discover in books & stationery at 9.6%. The grid is not additive: 29 cells sit more than ten points from what a category effect plus an intent effect predicts. On substitute tasks, 57.4% of failures are the agent giving up early.

5.5 Protocols change cost more than success

Raw success spans 60.8% to 72.2% across the ten paying cells, but raw rates mix the protocol with whichever tasks landed on it. We therefore fit a conditional logit with task fixed effects [29] on paying tasks whose outcome varies across cells (Figure 5). Against UCP over MCP with the UCP processor, ACP with delegated payment multiplies the odds of success by 1.17 [1.01, 1.36] and WebMCP with the UCP processor by 1.14 [1.01, 1.28]; MPP lowers them (0.82 [0.72, 0.92] over MCP, 0.85 over REST); the Shop Pay cells are too few to separate from one. AP2 mandates appear to multiply the odds by 3.12, but that edge is the judge’s: AP2’s final checkout reply lists no shipping options, so the fulfilment and deadline checks never apply there. On an outcome restricted to the checks every cell can fail, the AP2 odds ratio is 0.95 [0.83, 1.10]. Among cart-only journeys the surface barely matters (UCP over REST 1.03 [0.94, 1.13] against UCP over MCP).

The traces suggest why WebMCP and MPP move: a WebMCP page exposes one cart the agent edits in place, while

MPP adds a payment challenge the agent must answer. Cost does not follow the odds (Figure 6): WebMCP cart journeys use a median 50k prompt tokens against 157k over MCP for the same success, and the MPP cells use the most tokens for the lowest odds. Platform packs change little: success runs from 66.0% to 70.6% across the nine.

5.6 Weak stores are weak at the first search

The design factors explain 7.5% of the variance in success, the store 1.9%, and the task within its store 52.4%; the rest is chance. Of 24,549 tasks run at least three times, 29.3% both succeed and fail. The store’s share is small per journey because success is binary and the stand-ins differ in shape and platform rules rather than in the quality of data a merchant chooses to publish; across stores the spread is still wide, from 54.2% at the fifth-percentile store to 82.3% at the ninety-fifth (Figure 7). No store is named or ranked.

What the bottom quarter of stores share is a first search that misses: a fitting product in the first results in 80.9% of journeys against 85.3% in the top quarter, and searches returning nothing 12.3% of the time against 7.5%. Tool errors, latency and whether the store carries the decisive detail do not differ, and no platform pack holds more than 16.5% of the bottom quarter. On those stores the agent mostly stops rather than buys wrong: false refusals run 11.2% against 4.6%, journeys with no cart 5.8% against 2.3%, and checkouts left open 9.7% against 4.3%, while paying for the wrong item moves less (16.3% against 10.1%). This comparison is observational; we have not yet changed a store’s search and rerun its tasks.

5.7 What the data found wrong with the instrument

Right by the judge, wrong by the record. judge-1.0 rated 14,888 released journeys right whose transaction record disagreed; the release publishes them as wrong, with the reason (Section 4). The oracle falls under the same rule on 1,729 of its 9,732 full-data journeys, which is why its success is 82.2% and not 100%.

Tools that do not exist. In 38 journeys the agent called a tool that no surface defines, most often a way to ask the shopper.

A detector that cannot read a negation. The code check that flags a purchase claim on a journey with no order fires on “no order was placed”; the decision model disagreed in 99.8% of 1,870 firings. The check is marked broken and left out of the recommended score.

6 What a store can do

The findings point at four things a store controls. Each was measured here on stand-ins and can be measured on a real store with the same tasks and judge; the store comparison above is observational, and the next step is to change one lever and rerun.

- **Answer the first query.** The bottom quarter of stores differ from the top in how often the first search finds a fitting product and how often it returns nothing (Section 5.6).

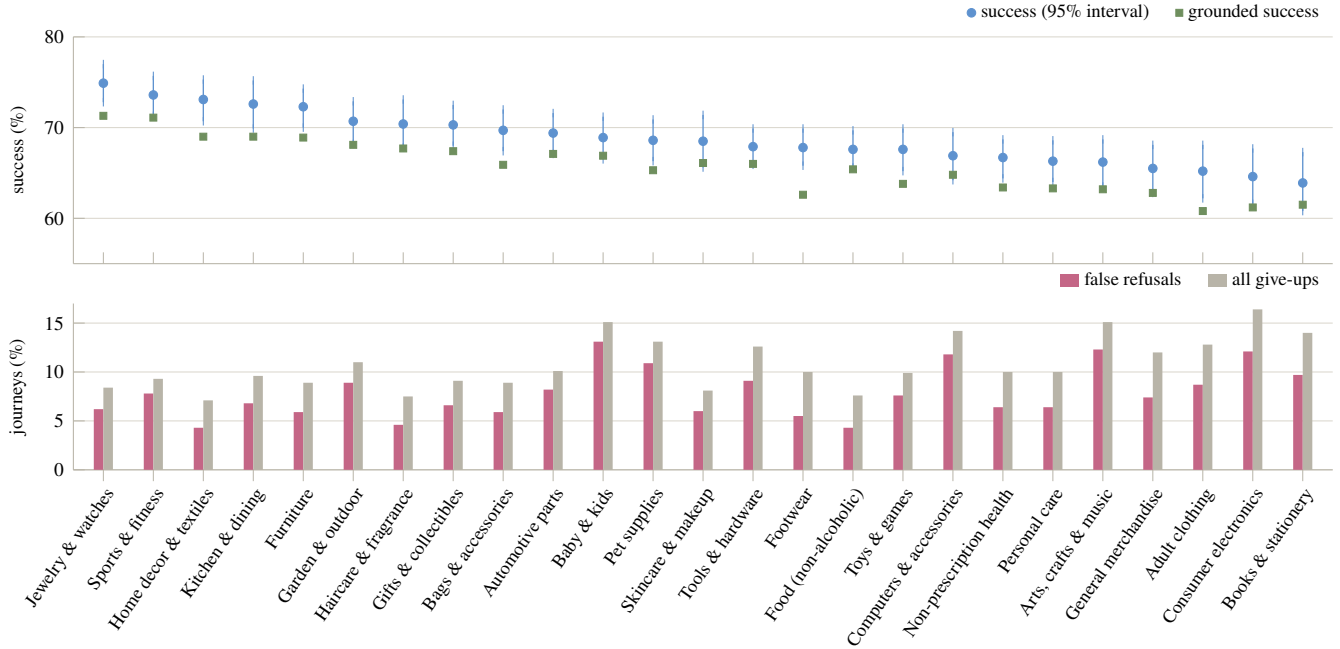


Figure 3. The 25 categories sorted by success. **Top:** success with store-resampled 95% intervals, and grounded success. **Bottom:** the share of journeys ending in a false refusal, and in any give-up (false refusals included). Appendix C gives every number.

- **Put the decisive facts where the agent reads first.** The agent carts from search results without opening a product in 37.6% of journeys, so options, stock and the deciding detail belong in the search result itself.
- **Serve substitutes and discovery.** They are the hard corner of every category (Section 5.4): in-stock alternatives for a named product, and facets an agent can walk.
- **Choose a protocol for what its checkout returns and what it costs,** not for its name (Section 5.5).

7 Limitations

One agent. One model agent ran, so the levels are Qwen3.8-27B’s, and we cannot say which contrasts hold for other agents. The replay kit runs any agent through the same worlds and judge.

Stand-in stores, not live ones. Nothing may be bought on a real store, so the stores are stand-ins built from open catalogs with synthetic prices and stock, and the release scores no real store. The sandbox also removes what makes live checkout hard: bot walls, logins, page changes and stores that are unreadable to agents to begin with [11]. Our near-zero checkout failure is a property of protocol calls in a sandbox, not a claim about the live web.

A programmed shopper. The shopper’s replies are narrower than a person’s, and it is built to pay: it never walks away, which simulated customers are known to overstate [6].

The judge. Outcomes rest on code and on the transaction record. Model-read checks are shadow annotations from a

decision model that shares the agent’s backbone; hand checks were labelled by study agents, and human-label rounds come before any question type is promoted. All numbers come from one judge version and one locked specification.

8 Release and intended use

CartTrace-100k is on Hugging Face as Parquet tables per journey: messages, episodes, turns, tool calls, wire exchanges, cart and checkout states, the sandbox’s event log, the agent’s prompt and tool definitions, the judge’s checks step by step with what settled each, the oracle’s journeys, a profile card per category, and a table that describes each stand-in store without scoring it. Splits hold out whole stores, task families and tasks. Store identities are pseudonyms; a leak scan over every string blocks the build on any secret, internal host, personal name or real store domain, and no trace of a real store ships. The replay kit rebuilds the sandbox and re-judges any journey. The dataset card states the licence of the data and of each source catalog, and the release carries a Croissant record [1].

The data is meant for studying and improving shopping agents and the stores and protocols they use: failure analysis, judge and verifier research, and training or evaluating agents on held-out stores. It is not a ranking of real merchants, platforms or protocol vendors, and its success rates should not be read as rates on the live web.

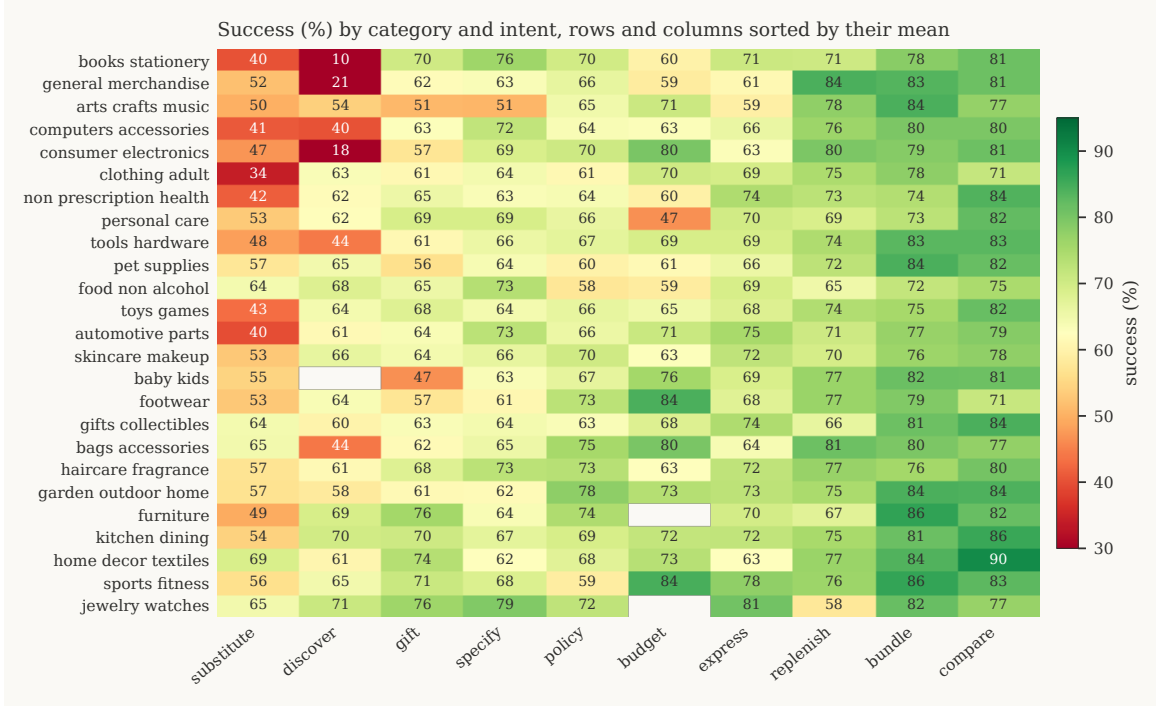


Figure 4. Success by category and intent, for cells with at least 50 journeys. Rows and columns are sorted by their mean, so the hard corner is top left.

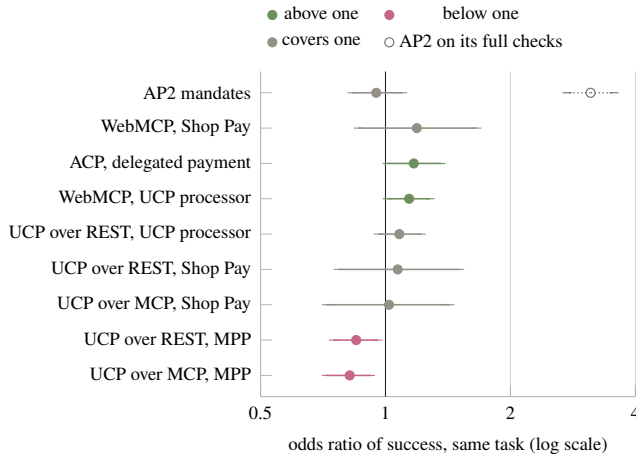


Figure 5. Paying journeys: odds of success for each surface and rail cell against UCP over MCP with the UCP processor, from a conditional logit with task fixed effects (95% intervals). AP2 is shown on the checks every cell can fail; its apparent edge on the full check set (hollow) comes from checks that never apply to AP2 (Section 5.5).

9 Conclusion

Across 25 categories, 1,044 stores and 13 protocol cells, one agent fails in the same two places: reading product facts and asking for the needs the shopper keeps back. Once the cart is right, checkout and payment succeed on every protocol. Substitute and discovery requests are where journeys are lost, and they are lost to giving up rather than to wrong purchases. The

protocol changes what a journey costs more than whether it succeeds. CartTrace-100k makes these journeys public with the judge’s checks beside them, so that the steps between a need and a paid order can be studied, re-judged, and run with other agents and new stores.

References

- [1] M. Akhtar, O. Benjelloun, C. Conforti, L. Foschini, J. Giner-Migueluez, P. Gijsbers, S. Goswami, N. Jain, M. Karamousadakis, M. Kuchnik, et al. Croissant: A Metadata Format for ML-Ready Datasets, 2024. URL <https://arxiv.org/abs/2403.19546>. NeurIPS 2024 Datasets and Benchmarks Track.
- [2] A. Allouah, O. Besbes, J. D. Figueroa, Y. Kanoria, and A. Kumar. What Is Your AI Agent Buying? Evaluation, Biases, Model Dependence, & Emerging Implications for Agentic E-Commerce, 2025. URL <https://arxiv.org/abs/2508.02630>.
- [3] A. Aviv, P. A. Gandh, R. Bitton, and A. Shabtai. Beyond the Mandate: A Systematic Security Analysis of the Agent Payments Protocol (AP2), 2026. URL <https://arxiv.org/abs/2608.23858>.
- [4] G. Bansal, W. Hua, Z. Huang, A. Fournay, A. Swearngin, W. Epperson, T. Payne, J. M. Hofman, B. Lucier, C. Singh, et al. Magentic Marketplace: An Open-Source Environment for Studying Agentic Markets, 2025. URL <https://arxiv.org/abs/2510.25779>.
- [5] V. Barres, H. Dong, S. Ray, X. Si, and K. Narasimhan. τ^2 -Bench: Evaluating Conversational Agents in a

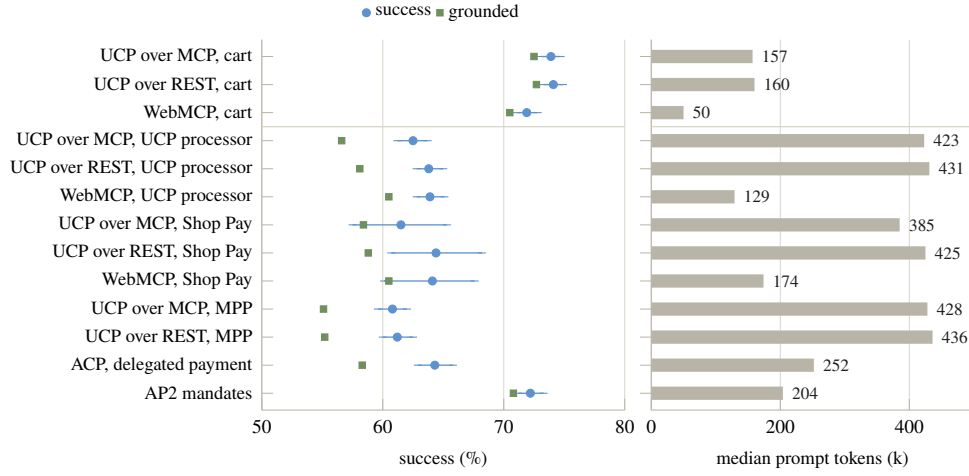


Figure 6. The thirteen surface and rail cells; the three cart-only cells sit above the line. **Left:** success with store-resampled 95% intervals, and grounded success. **Right:** median prompt tokens per journey.

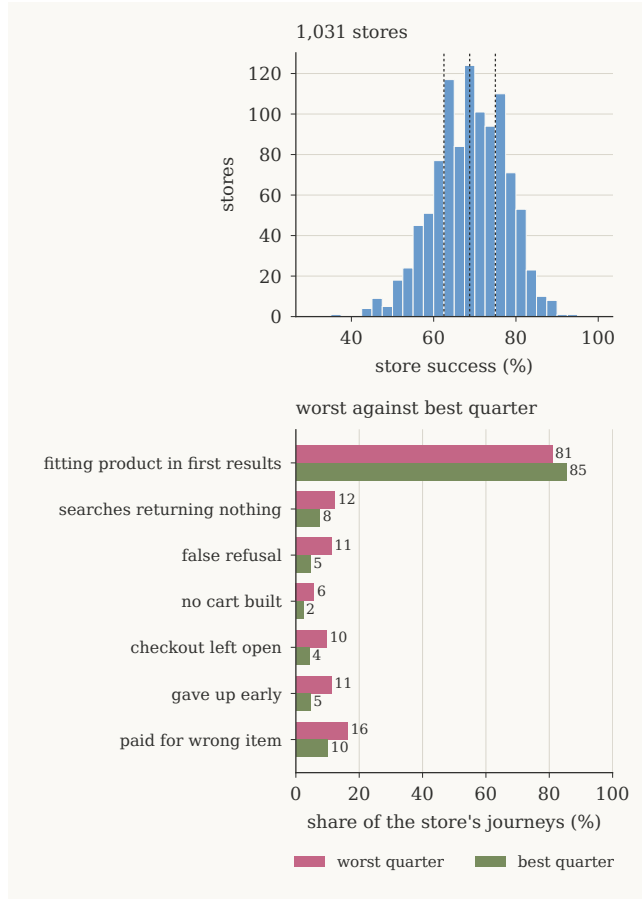


Figure 7. Top: success per stand-in store (1,031 stores with at least 60 full-data journeys), with quartile cuts. Bottom: the worst quarter of stores against the best on what search gave the agent and how journeys ended.

Dual-Control Environment, 2025. URL

<https://arxiv.org/abs/2506.07982>.

- [6] L. Chen. Simulated Customers Never Walk Away: Decision Fidelity of LLM User Simulators Measured Against Real Purchase Outcomes, 2026. URL <https://arxiv.org/abs/2606.20708>.
- [7] Y. Chen, S. Liu, Z. Liu, W. Sun, L. Baltrunas, and B. Schroeder. WANDS: Dataset for Product Search Relevance Assessment. In *Advances in Information Retrieval (ECIR 2022)*, Lecture Notes in Computer Science, pages 128–141. Springer, 2022. doi: 10.1007/978-3-030-99736-6_9. Data: <https://github.com/wayfair/WANDS> (MIT licence).
- [8] M. Cherep, C. Ma, A. Xu, M. Shaked, P. Maes, and N. Singh. A Framework for Studying AI Agent Behavior: Evidence from Consumer Choice Experiments. In *International Conference on Learning Representations (ICLR)*, 2026. URL <https://arxiv.org/abs/2509.25609>.
- [9] Cloudflare. Introducing Clef: our open-source decision models, and new RL fine-tuning platform, 2026. URL <https://blog.cloudflare.com/clef-decision-models/>. Blog post, 2026-10-01. Weights: huggingface.co/Cloudflare/clef and [Cloudflare/clef-flash](https://huggingface.co/Cloudflare/clef-flash) (Apache-2.0).
- [10] J. Collins, S. Goel, K. Deng, A. Luthra, L. Xu, E. Gundogdu, X. Zhang, T. F. Y. Vicente, T. Dideriksen, H. Arora, M. Guillaumin, and J. Malik. ABO: Dataset and Benchmarks for Real-World 3D Object Understanding, 2021. URL <https://arxiv.org/abs/2110.06199>.
- [11] A. de Valois-Franklin, A. Bogdan, A. Ibrahim, I. Shaer, and T. Andguladze. Get In, Decide, Prove: A Merchant-Side Readiness Benchmark for Agentic Commerce, 2026. URL <https://arxiv.org/abs/2610.17378>.
- [12] T. Debi, W. Zhu, and P. Sen Gupta. Whispers of Wealth: Red-Teaming Google’s Agent Payments Protocol via Prompt Injection. In *International Conference on Intelligent Multimedia, Networking, and Security (IMNS)*, 2026. URL <https://arxiv.org/abs/2601.22569>.
- [13] Y. Dou, M. Galley, B. Peng, C. Kedzie, W. Cai, A. Ritter,

- C. Quirk, W. Xu, and J. Gao. SimulatorArena: Are User Simulators Reliable Proxies for Multi-Turn Evaluation of AI Assistants?, 2025. URL <https://arxiv.org/abs/2510.05444>. EMNLP 2025.
- [14] Z. Du, T. Li, and H. Zhang. EComAgentBench: Benchmarking Shopping Agents on Long-Horizon Tasks with Distributed Hidden Intent, 2026. URL <https://arxiv.org/abs/2606.17698>. EMNLP 2026 Industry Track.
- [15] S. Fan, M. Yang, D. Wang, C. Chen, Y. Zhang, H. Wei, M. Li, J. McAuley, K. Zhang, P. S. Yu, K. Yu, and Z. Liu. Agentic Commerce World: An Auditable and Verifiable Environment for Vibe Commerce, 2026. URL <https://arxiv.org/abs/2608.02441>.
- [16] I. Finder, A. Elovic, G. Shalev, and L. Yosef. AX is the New AEO, 2026. URL <https://arxiv.org/abs/2609.34951>.
- [17] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé, and K. Crawford. Datasheets for Datasets, 2018. URL <https://arxiv.org/abs/1803.09010>. Communications of the ACM, 2021.
- [18] Google. Agent Payments Protocol (AP2), 2025. URL <https://github.com/google-agentic-commerce/AP2>.
- [19] K. Jiang, M. Yu, Y. Chang, M. K. Jangid, J. Niu, C. Wang, and Y. Zhang. A Formal Analysis of Agent Payment Protocols, 2026. URL <https://arxiv.org/abs/2609.00060>.
- [20] J. Jung, F. Brahman, and Y. Choi. Trust or Escalate: LLM Judges with Provable Guarantees for Human Agreement, 2024. URL <https://arxiv.org/abs/2407.18370>.
- [21] S. Kim, R. Heo, Y. Seo, J. Yeo, and D. Lee. AgenticShop: Benchmarking Agentic Product Curation for Personalized Web Shopping, 2026. URL <https://arxiv.org/abs/2602.12315>. WWW 2026.
- [22] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention, 2023. URL <https://arxiv.org/abs/2309.06180>. SOSP 2023.
- [23] Y. Li, Y. Miao, R. Krishnan, and R. Padman. JEV-as-a-Judge: Accept When Confident, Escalate When Unsure, 2026. URL <https://arxiv.org/abs/2609.26550>.
- [24] Y. Louck. Protocol-Level Attacks on Agentic Commerce Platforms: A Cross-Platform Taxonomy, AIP-Bench, and Unified Defense, 2026. URL <https://arxiv.org/abs/2607.21824>.
- [25] Y. Louck, A. Dvir, and A. Stulman. Signing the Transaction but Not the Decision: Whisper Attacks and a Binding Defense for AP2, 2026. URL <https://arxiv.org/abs/2609.11757>.
- [26] X. H. Lù, G. Kamath, M. Mosbach, and S. Reddy. Build the Web for Agents, Not Agents for the Web, 2025. URL <https://arxiv.org/abs/2506.10953>.
- [27] Y. Lyu, X. Zhang, L. Yan, M. de Rijke, Z. Ren, and X. Chen. DeepShop: A Benchmark for Deep Research Shopping Agents, 2025. URL <https://arxiv.org/abs/2506.02839>.
- [28] X. H. Lù, A. Kazemnejad, N. Meade, A. Patel, D. Shin, A. Zambrano, K. Stańczak, P. Shaw, C. J. Pal, and S. Reddy. AgentRewardBench: Evaluating Automatic Evaluations of Web Agent Trajectories, 2025. URL <https://arxiv.org/abs/2504.08942>.
- [29] D. McFadden. Conditional logit analysis of qualitative choice behavior. In P. Zarembka, editor, *Frontiers in Econometrics*, pages 105–142. Academic Press, 1974.
- [30] E. Miller. Adding Error Bars to Evals: A Statistical Approach to Language Model Evaluations, 2024. URL <https://arxiv.org/abs/2411.00640>.
- [31] Model Context Protocol contributors. Model Context Protocol Specification, 2024. URL <https://modelcontextprotocol.io/specification/latest>. Specification version 2026-07-28; introduced by Anthropic.
- [32] T. Naous, P. Laban, W. Xu, and J. Neville. Flipping the Dialogue: Training and Evaluating User Language Models, 2025. URL <https://arxiv.org/abs/2510.06552>. ICLR 2026.
- [33] OpenAI and Stripe. Agentic Commerce Protocol (ACP), 2025. URL <https://github.com/agentic-commerce-protocol/agentic-commerce-protocol>. Open specification maintained by OpenAI and Stripe, status beta.
- [34] A. Panickssery, S. R. Bowman, and S. Feng. LLM Evaluators Recognize and Favor Their Own Generations, 2024. URL <https://arxiv.org/abs/2404.13076>.
- [35] R. Peeters, A. Steiner, L. Schwarz, J. Y. Caspary, and C. Bizer. WebMall – A Multi-Shop Benchmark for Evaluating Web Agents, 2025. URL <https://arxiv.org/abs/2508.13024>. SIGIR 2026.
- [36] C. Qian, Z. Liu, A. Prabhakar, Z. Liu, J. Zhang, H. Chen, H. Ji, W. Yao, S. Heinecke, S. Savarese, C. Xiong, and H. Wang. UserBench: An Interactive Gym Environment for User-Centric Agents, 2025. URL <https://arxiv.org/abs/2507.22034>.
- [37] Qwen Team. Qwen3.8-72B, 2026. URL <https://huggingface.co/Qwen/Qwen3.8-72B>. Model card, Apache-2.0. Recommended citation: Qwen Team, “Qwen3.8-Max: A New Bar for Coding and Cowork”, August 2026, <https://qwen.ai/blog?id=qwen3.8>.
- [38] A. Rafe and S. Das. Benchmarking System One decision models against trained classifiers and language models for automated decision gates, 2026. URL <https://arxiv.org/abs/2610.00346>.
- [39] I. Saracay, L. Schmidt, and C. Guestrin. Beyond Expert Users: Agents Should Help Users Construct Preferences, Not Just Elicit Them, 2026. URL <https://arxiv.org/abs/2606.30863>.
- [40] N. Shah and H. Sáez de Ocariz Borde. Evaluating Open-Weight E-Commerce Agents with Environment-Grounded Verification, 2026. URL <https://arxiv.org/abs/2609.16093>.
- [41] R. Sharma, A. B. Duncan, G. Marceau Caron, and S. J. Vollmer. TypedBench: A Benchmark for Calibration, Framing Sensitivity, and Cost in System One Decision Models, 2026. URL <https://arxiv.org/abs/2610.11392>.
- [42] L. Shi, C. Ma, W. Liang, X. Diao, W. Ma, and S. Vosoughi. Judging the Judges: A Systematic Study of Position Bias in LLM-as-a-Judge, 2024. URL <https://arxiv.org/abs/2406.07791>. AACL-IJCNLP 2025.
- [43] A. Srivastava and D. Paul. Agentic Commerce Bench:

- Measuring Fraud Detection for Agents That Spend Money, 2026. URL <https://arxiv.org/abs/2609.35886>.
- [44] A. Steiner, R. Peeters, and C. Bizer. MCP vs RAG vs NLWeb vs HTML: A Comparison of the Effectiveness and Efficiency of Different Agent Interfaces to the Web. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, pages 8493–8496, 2026. URL <https://arxiv.org/abs/2511.23281>.
- [45] Stripe. How Stripe Is Designing Checkout for AI Agents. Stripe engineering blog, 22 September 2026, 2026. URL <https://stripe.dev/blog/how-stripe-is-designing-checkout-for-ai-agents>.
- [46] Tempo and Stripe. Machine Payments Protocol (MPP), 2026. URL <https://mpp.dev/>. Open standard co-authored by Tempo and Stripe; site repository <https://github.com/tempoxyz/mpp>; Python SDK pympp 0.11.0 on PyPI.
- [47] U. Uchibeke. APort Vault: Benchmarking AI Agent Payment Authorization with the Open Agent Passport, 2026. URL <https://arxiv.org/abs/2609.22076>.
- [48] UCP contributors. Universal Commerce Protocol (UCP), 2026. URL <https://ucp.dev/>. Specification repository: <https://github.com/Universal-Commerce-Protocol/ucp> (Apache-2.0).
- [49] J. Wang, K. Xiao, Q. Sun, H. Zhao, T. Luo, J. D. Zhang, and X. Zeng. ShoppingBench: A Real-World Intent-Grounded Shopping Benchmark for LLM-based Agents, 2025. URL <https://arxiv.org/abs/2508.04266>. AAAI 2026.
- [50] Web Machine Learning Community Group. WebMCP. W3C Draft Community Group Report, version of 9 October 2026, 2026. URL <https://webmachinelearning.github.io/webmcp/>. Explainer and issues: <https://github.com/webmachinelearning/webmcp>.
- [51] T. Xue, W. Qi, T. Shi, C. H. Song, B. Gou, D. Song, H. Sun, and Y. Su. An Illusion of Progress? Assessing the Current State of Web Agents, 2025. URL <https://arxiv.org/abs/2504.01382>. COLM 2025.
- [52] S. Yao, H. Chen, J. Yang, and K. Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents, 2022. URL <https://arxiv.org/abs/2207.01206>. NeurIPS 2022.
- [53] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains, 2024. URL <https://arxiv.org/abs/2406.12045>.
- [54] Z. Yu, K. Xiao, H. Zhao, T. Luo, and X. Zeng. Shopping Companion: Benchmarking and Training LLM Agents for Long-Horizon Preference-Grounded E-Commerce Tasks, 2026. URL <https://arxiv.org/abs/2603.14864>.
- [55] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena, 2023. URL <https://arxiv.org/abs/2306.05685>. NeurIPS 2023 Datasets and Benchmarks Track.
- [56] Y. Zhu, T. Jin, Y. Pruksachatkun, A. Zhang, S. Liu, S. Cui, S. Kapoor, S. Longpre, K. Meng, R. Weiss, et al. Establishing Best Practices for Building Rigorous Agentic Benchmarks, 2025. URL <https://arxiv.org/abs/2507.02825>.

Table 3. The judge’s ten stages on Qwen3.8-27B full-data journeys: questions in the stage, the share of applicable steps settled by code and by the decision model, and the share of journeys whose counted checks at the stage score below one half, for journeys that stop at the cart and journeys that pay.

Stage (%)	Qs	Code	Model	Fail cart	Fail pay
understand	5	72.7	25.9	0.0	0.1
discover	9	85.2	14.7	0.3	0.3
read facts	8	99.8	0.2	37.8	37.3
clarify	5	72.4	25.5	28.8	29.0
choose variant	10	80.3	18.8	8.0	8.0
build cart	7	100.0	0.0	0.2	0.1
check out	13	92.0	8.0	0.0	0.0
pay	7	75.0	24.9	–	0.0
confirm	6	87.3	12.4	15.7	4.7
report	13	62.2	36.0	0.0	0.6

A Glossary

Stand-in store A sandbox store with a real store’s category, market and shape, serving open-catalog products.

Platform pack The rules of one e-commerce platform (search, discounts, shipping, refused lines, MCP handshake) under which a stand-in serves its products.

Surface How the agent reaches the store: UCP over MCP, UCP over REST, WebMCP, ACP over REST, or AP2.

Rail How a paying journey pays: UCP processor, Shop Pay, ACP delegated payment, AP2 mandates, or MPP.

Cell One surface–rail pair, or a surface alone for cart-only journeys; 13 in all.

Hidden specification The predicates a right cart must satisfy; each is stated, kept until asked, or defaulted.

Full data A satisfiable task with a persona built to pay. All headline results use full-data journeys.

Designed failure An impossible task, a payment fault or a checkout fault, scored apart from full data.

Success A right cart or order (including right by luck) or a correct refusal, after each verdict is held to the store’s transaction record (labels 1.1).

Grounded success Success without right-by-luck, and with every claim the agent made backed by what it saw.

Oracle A scripted agent that reads the hidden specification; the ceiling of what world and labels allow.

B The judge’s ten stages

Table 3 gives each stage’s question count, who settles its steps, and the share of journeys failing it, separately for journeys that stop at the cart and journeys that pay.

C Every category

Table 4 gives every number behind Figure 3.

D Intent

Discover a need without a product name. **Compare** two or more candidates. **Specify** an exact variant. **Substitute** the

Table 4. Every category, Qwen3.8-27B full-data journeys, sorted by success. Intervals resample whole stores (95%). Give-ups include false refusals. Turns and tokens are medians per journey.

Category	Journeys	Stores	Success (%)	Grounded	False ref.	Give-up	Turns	Tokens
Jewelry & watches	3,879	41	74.9 [72.8, 77.0]	71.3	6.2	8.4	41	230,461
Sports & fitness	3,872	42	73.6 [71.3, 75.7]	71.1	7.8	9.3	37	184,265
Home decor & textiles	3,901	42	73.1 [70.7, 75.3]	69.0	4.3	7.1	37	198,989
Kitchen & dining	3,909	42	72.6 [69.9, 75.2]	69.0	6.8	9.6	37	188,271
Furniture	3,958	42	72.3 [70.0, 74.3]	68.9	5.9	8.9	39	206,260
Garden & outdoor	3,870	41	70.7 [68.2, 72.9]	68.1	8.9	11.0	38	201,196
Haircare & fragrance	3,949	43	70.4 [67.8, 73.1]	67.7	4.6	7.5	38	206,658
Gifts & collectibles	3,927	42	70.3 [68.0, 72.5]	67.4	6.6	9.1	38	204,212
Bags & accessories	3,933	43	69.7 [67.4, 72.0]	65.9	5.9	8.9	40	225,620
Automotive parts	3,873	42	69.4 [67.1, 71.6]	67.1	8.2	10.1	36	168,426
Baby & kids	3,873	43	68.9 [66.5, 71.2]	66.9	13.1	15.1	38	206,506
Pet supplies	3,918	41	68.6 [66.3, 70.9]	65.3	10.9	13.1	36	185,924
Skincare & makeup	3,912	41	68.5 [65.6, 71.4]	66.1	6.0	8.1	38	194,456
Tools & hardware	3,877	41	67.9 [65.9, 69.9]	66.0	9.1	12.6	38	178,122
Footwear	3,855	41	67.8 [65.8, 69.9]	62.6	5.5	10.0	42	266,573
Food (non-alcoholic)	3,916	41	67.6 [65.6, 69.7]	65.4	4.3	7.6	38	179,927
Toys & games	3,907	41	67.6 [65.2, 69.9]	63.8	7.6	9.9	39	224,016
Computers & accessories	3,869	41	66.9 [64.2, 69.5]	64.8	11.8	14.2	37	196,222
Non-prescription health	3,882	42	66.7 [64.4, 68.7]	63.4	6.4	10.0	38	196,505
Personal care	3,938	42	66.3 [64.0, 68.6]	63.3	6.4	10.0	37	185,089
Arts, crafts & music	3,896	42	66.2 [63.5, 68.7]	63.2	12.3	15.1	39	188,431
General merchandise	3,860	41	65.5 [62.8, 68.1]	62.8	7.4	12.0	40	184,475
Adult clothing	3,884	42	65.2 [62.2, 68.1]	60.8	8.7	12.8	42	259,902
Consumer electronics	3,867	44	64.6 [61.6, 67.7]	61.2	12.1	16.4	40	210,809
Books & stationery	3,907	41	63.9 [60.8, 67.3]	61.5	9.7	14.0	39	183,304

named product is out of stock and a similar one will do. **Bundle** several items in one order. **Gift** for someone else. **Replenish** a repeat purchase. **Express** fast delivery. **Budget** a hard price ceiling. **Policy** a question about returns, shipping or terms before buying.

E The agent’s instructions

Every journey uses one system prompt. Its rules: read candidates before choosing; ask one question at a time, and only when a real choice is open; enter at checkout only what the shopper gave; say what is in the order before the wallet step; never retry a declined payment unless asked. The full prompt and tool definitions ship with every journey in the release.

F Reproducibility and accounting

The official run produced 111,749 bundles. Release v1 excludes 1,145 (quarantined reruns, duplicates and rows of other models) and publishes 101,416 Qwen3.8-27B journeys and 10,132 oracle journeys. 8 H100 GPUs served the agent for the 121 hours of the run (971 GPU hours); summed model-call latency over the released journeys is 5,355 hours because calls overlapped. The judge ran on every bundle, and its gate passed 101,408 Qwen3.8-27B journeys, the denominator throughout. The shopper’s fallback reply is blamed for 6.2% of judged journeys and the harness for none.

G An example journey

Table 5 shows a paid journey from the release (arts, crafts & music; intent express; AP2 mandates), abridged to messages and tool calls. Store and order identifiers are the release’s pseudonyms.

Table 5. An example paid journey from the release (episode ep-8df908fb65e4), abridged to messages and tool calls.

shopper	I need a ukulele and I’m in a hurry. Any good one will do. I’d rather collect it in store.
search_catalog	“ukulele” → 4 products
get_product	→ product details
agent	I found the store’s only ukulele: a 4-string mini ukulele, \$27.99, in stock and available for in-store pickup. Since you said any good one would do, I’ll set that up.
send_intent_mandate	constraints → 1 product
create_checkout	intent mandate, items, shipping option → incomplete; again with payer email → ok
agent	The checkout is ready: 1 mini ukulele (light brown), \$27.99 + \$2.24 tax = \$30.23, free in-store pickup at the store counter. Please review and approve in the wallet.
wallet	Sign on your device: pay \$30.23. Confirm?
shopper	Yes, go ahead and place the order for \$30.23.
create_checkout_mandate	→ signed
issue_payment_credential	→ ok
complete_checkout	→ success; get_checkout_status → success
agent	Order placed: 1 mini ukulele (light brown, 4-string), total \$30.23 including tax. Payment succeeded. It’s ready for in-store pickup at the store counter.