Token-Aware Workload Persona Clustering for Browser Shopping Agents: Lightweight Capacity Modeling from Web Interaction Trajectories

Jiayi Nie[1], Yuxuan Ren[2], Jessica Chen[3]
[1] *Operations Research, Columbia University, NY, USA*
[2] *Chemical Engineering & Data Science, University of Washington, WA, USA*
[3] *Data Science, Columbia University, NY, USA*
renmichelleyuxuan@gmail.com

Keywords

browser agents; WebShop; MiniWoB++; workload clustering; token-aware capacity planning;

Abstract

Browser agents are usually evaluated by task success, reward, or human preference, yet the same trajectories also determine the serving load imposed on a language-model system. This paper studies browser shopping work as a capacity-planning problem. Each WebShop trajectory is transformed into a workload unit with instruction tokens, observation-token proxies, action counts, page types, candidate action counts, reward, and derived latency-risk indicators. The study uses 1,643 WebShop human trajectories, a 1,000-product WebShop catalog subset, 12,251 crowd instructions, and a deterministic MiniWoB++ validation sample of 1,467 demonstrations across 60 task directories. KMeans clustering over post-hoc trajectory features yields four operational personas: Direct buyers, Option comparators, Catalog scanners, and Recovery browsers. Direct buyers make up 73.5% of trajectories and require 637 prompt-token units on average, while the rare Catalog scanner persona uses 9,499 prompt-token units and 73 actions on average. Planning-time token prediction remains difficult when only instruction and product fields are available: Ridge regression achieves a held-out RMSE of 1,328.7 token units, only slightly ahead of the mean baseline, while logistic high-cost detection reaches AUC 0.688 and F1 0.503. A queue-based capacity simulation shows that under the observed persona mix, a four-worker service lane remains stable through 50 tasks per minute but crosses 38.1% wait-over-two-second risk at 60 tasks per minute. The findings show that browser-agent workloads have heavy-tailed serving costs, and that lightweight persona metadata can expose capacity pressure that is invisible in task-success metrics alone.

Introduction

Autonomous browser agents join language understanding with concrete actions over webpages. In online shopping settings, an agent must understand a natural-language request, search for products, inspect result pages, open item pages, select options, and decide when to buy [26], [27]. WebShop made this setting measurable by providing a simulated e-commerce site with real products and crowd instructions, while earlier web-interaction environments such as World of Bits and MiniWoB focused on low-level browser tasks and reproducible web rewards [1], [2], [3]. More recent language-agent work has shown that web use, search, reasoning, and action can be combined in browser-assisted question answering and ReAct-style interleaved reasoning-action traces [4], [5]. These lines of work have greatly improved the ability to evaluate whether an agent reaches a goal.

Task success, however, is not the only quantity that matters when browser agents are deployed through large language models. A shopping task that succeeds in four clicks and a task that succeeds after seventy result-page visits are different serving workloads even if their final rewards match. Transformer language models scale costs with input and generated tokens, while model families such as GPT-3 [6], PaLM [7], and OPT [8] have made long-

context inference a first-order systems problem on top of Transformer architectures [9]. Serving systems for autoregressive Transformers also show that latency and throughput depend on request shape, batching behavior, and token-generation iteration schedules [10], [28]. Browser-agent trajectories therefore deserve a second reading: they are not only behavioral traces, but also capacity-demand traces.

This paper develops that second reading for browser shopping agents. The central idea is simple. Each trajectory is treated as a workload unit whose service demand can be summarized by token proxies, action counts, candidate-action exposure, page-type mix, and backtracking behavior. These features are then used in three connected analyses: persona clustering, prediction of token demand from planning-time features, and simulation of capacity pressure under different arrival rates. The resulting view complements reward-based evaluation. It answers questions such as: Which kinds of shopping tasks create the longest prompts? Which trajectory personas dominate capacity risk? Can instruction and product metadata detect high-cost tasks before a full trajectory unfolds? How much under-provisioning appears when the observed persona mix arrives at higher request rates?

A token-aware view is also useful because browser-agent workloads combine two sources of uncertainty. The first is linguistic: users describe desired products with different numbers of attributes, prices, colors, sizes, and implicit constraints. The second is environmental: the same request can expose dense result pages, ambiguous item titles, option menus, or product pages whose text does not match the instruction cleanly. A serving system sees both sources as prompt growth and additional action turns. Separating these sources helps distinguish a genuinely complex user request from a simple request that becomes expensive because the catalog is noisy or the search path recovers from earlier choices.

The paper does not argue that token metrics should replace reward. Reward remains the principal measure of whether a shopping agent solved the user's task. The claim is that reward is incomplete for deployment. A high-reward trajectory can still create large context windows, repeated page observations, and long waits for other users sharing the same serving lane. Conversely, a short failed trajectory may be cheap but unhelpful. Reporting both quality and workload gives a more complete picture of whether a browser agent is ready for production-style use [29], [30].

The study is intentionally lightweight. It does not require training a new browser agent or running a large model. Instead, it reuses human interaction trajectories and product/instruction fields to build empirical workload statistics. This choice makes the experiments repeatable on modest hardware and aligns with a practical capacity-planning workflow: first estimate workload structure from logs, then decide where a serving stack needs larger context windows, conservative batching, or a separate high-cost lane. The work is also deliberately scoped to browser shopping agents rather than general browser control. Shopping tasks provide rich search, selection, option, and purchase actions, while keeping page types interpretable enough for operational analysis.

A capacity interpretation changes how browser-agent logs are used. In a conventional benchmark table, two trajectories with the same final reward are often treated as equivalent. In a serving system, they are not equivalent: one trajectory may create a short prompt, a single result page, and one buy action, while another may repeatedly expand search results, open near-matching items, inspect details, and backtrack. The latter trajectory consumes context budget, attention computation, and scheduler time even when it eventually succeeds. This paper therefore treats each completed trace as a small performance profile rather than only as an outcome label.

The workload-persona framing also supports practical communication between model builders and infrastructure engineers. Model builders often describe a failure as a search, grounding, or option-selection problem. Infrastructure engineers often describe the same episode as a long-context request, a queueing outlier, or a batching hazard. Personas connect these views by giving names to repeated workload shapes. A Direct buyer is a low-risk trace because it keeps the page sequence narrow. A

Catalog scanner is risky because it repeatedly exposes large candidate sets. A Recovery browser is risky because backtracking and repeated state changes accumulate prompt tokens. These labels make capacity pressure interpretable without requiring a new benchmark or a new agent architecture [31], [32].

The contributions are fourfold. First, the paper defines a trajectory-to-workload transformation for WebShop that includes token proxies, action counts, candidate-action counts, page-type counts, and reward. Second, it derives four workload personas that separate direct purchases from option comparison, catalog scanning, and recovery-heavy browsing. Third, it evaluates planning-time regression and classification models for token demand and high-cost detection. Fourth, it connects the personas to a queue-based capacity simulation and to compact task-level capacity memos [33]. The same extraction logic is also checked against MiniWoB++ demonstrations to verify that action and token workload variation appears beyond the shopping domain.

Method

The experimental unit is a complete browser trajectory. For WebShop, a trajectory file records the page sequence, the goal object, page content fields, terminal reward when present, and selected item/options. The feature extractor first reads every trajectory and infers action types from page transitions. For example, an index-to-results transition is a search action, a results-to-item transition is an item-choice action, an item-to-item transition with changed options is an option or variant action, and an item-to-done transition is a buy action. This transition-based action reconstruction is appropriate for the available trajectory format because the files store page states rather than explicit click events.

Token demand is approximated with a lexical token proxy rather than a proprietary tokenizer. The instruction token count is the number of word and punctuation tokens in the user request. The observation token proxy is computed from the page and content fields visible at each state. The total prompt-token proxy sums the instruction tokens, current observation tokens, and a fixed action-schema allowance of twenty tokens for every non-terminal step. This proxy is not intended to replace a tokenizer used by a specific model; it is an architecture-independent workload measure that preserves relative differences among short, medium, and long interaction traces. Product-text tokens are computed from the goal product name, query, category path, attributes, and options. Candidate-action count is the sum of visible search results, navigation buttons, item-page controls, option slots, and terminal buy choices across the trajectory.

The main WebShop corpus contains 1,643 human trajectories. The product-side feature check uses the 1,000-product subset, and the instruction-side check counts 12,251 crowd instructions. MiniWoB++ is used as an external validation source. Because the demonstration archive is much larger than the shopping trajectory corpus, the validation sample is deterministic and task-stratified: up to twenty-five demonstrations are selected from each task directory, giving 1,467 parsed demos across 60 task names. The MiniWoB++ features use utterance tokens, action counts, state counts, file-token proxies, tag counts, click/key action counts, and reward.

Persona clustering uses standardized trajectory features: instruction tokens, total prompt-token proxy, trajectory steps, candidate-action count, counts of search, item, and detail pages, unique item ASINs, option changes, backtracks, product-text tokens, and reward. KMeans is used for compact operational personas following the classic partitioning formulation of MacQueen [11]. PCA provides the two-dimensional visualization of the resulting clusters [12]. Non-linear embedding and density methods such as UMAP and DBSCAN/HDBSCAN are natural alternatives for larger or more irregular logs [13], [14], but the goal here is an interpretable capacity taxonomy with a fixed number of lanes.

The page taxonomy is intentionally small. WebShop pages are mapped to index, search-results, item-page, item-detail, and terminal states. This is enough to separate search expansion from product inspection and option handling. Search-result pages contribute candidate actions through result ASINs

and navigation controls. Item pages contribute action candidates through item controls, option slots, detail-page links, and the buy action. Item-detail pages add smaller candidate sets but signal deeper evidence gathering. The compact taxonomy avoids assumptions about a specific browser renderer while preserving the decision structure of shopping tasks.

The high-cost label is defined from the empirical distribution rather than an arbitrary fixed threshold. The top quartile of total prompt-token proxy is labeled high cost, which yields a threshold of 1,354.5 token units in the WebShop trajectory corpus. This definition is useful for routing because the positive class always represents the tasks most likely to stress the local serving lane [34], [35]. In another deployment, the same pipeline can replace the quartile cutoff with a service-level objective, a context-window budget, or a measured latency threshold.

For token demand prediction, the target is total prompt-token proxy. The predictors are planning-time or early fields: instruction tokens, first-query tokens, attribute count, option count, price-constraint flag, price upper bound, product-text tokens, and product category. Ridge regression provides a linear baseline with shrinkage [15]. Random Forest regression provides a nonlinear comparison [16]. The split is a deterministic 70/30 train-test split. The high-cost detector uses the same predictor set and labels the top quartile of total prompt-token proxy as high cost. Logistic regression supplies an interpretable binary classifier [17], and Random Forest classification provides a nonlinear comparison. AUC, F1, precision, recall, and accuracy are reported following standard ROC and classification-measure practice [18], [19].

Feature extraction follows a conservative parsing rule. When a field is missing, the extractor records zero for that field rather than inferring content from another product or from a later state. When a page contains several text fields, the proxy counts their concatenated lexical tokens once for that state. When the same item is revisited, the revisit still contributes observation tokens because a language-model agent would normally receive a fresh state description after the action. This rule intentionally favors serving-cost accounting over semantic deduplication: repeated exposure to the same information can still consume context and attention in a live prompt.

The train-test protocol separates planning-time features from realized trajectory features. Planning-time features are those available before or at the first search step: instruction length, constraint counts, product-side text, category, and first-query length when present. Realized features such as total steps, backtracks, number of item pages, and total candidate actions are withheld from the prediction models because they would not be known at admission time. They are used for clustering because personas are intended to summarize completed workload behavior, not to serve as pre-execution labels. This separation keeps the prediction task operationally honest while still allowing post-hoc personas to guide monitoring and lane design.

The product catalog subset is used as a consistency check on product-text scale. Product descriptions, titles, categories, ratings, availability text, and option groups are tokenized with the same proxy used for trajectory fields. This confirms that product pages have enough textual variation to affect prompt size even before a trajectory begins. The human-instruction file is used to check that the language distribution is not limited to the 1,643 trajectories alone. Together, the trajectory, product, and instruction files support a workload view that links user language, catalog text, and observed browser actions.

No model-generated trajectories are added. The empirical results are computed from the WebShop human trajectories and the MiniWoB++ demonstration files. This matters because workload personas should reflect real interaction behavior, including inefficient search, backtracking, and option correction. A synthetic policy could smooth away those behaviors and produce an unrealistically clean cost distribution.

Capacity simulation turns persona-level workload into a queueing exercise. For each simulated task, the persona is sampled from the observed persona distribution. Token demand is sampled from that persona's empirical mean and variance, lower-

bounded at fifty token units. The service-time proxy is token_proxy/500 + 0.08*trajectory_steps seconds on each of four identical workers. Arrivals follow a Poisson process over a 180-minute horizon. The reported pressure measures are mean wait, p95 wait, utilization, and the fraction of tasks waiting more than two seconds. The simulation follows standard queueing and system-performance modeling practice [24], [25]. The exact service-time calibration is a lightweight capacity unit, not a claim about a particular commercial model or GPU.

Implementation used scikit-learn estimators [20], pandas data frames [21], NumPy arrays [22], and Matplotlib figures [23]. All random choices use seed 42. The code writes feature tables before modeling, then writes every metric table and figure from those saved features. This separation makes it possible to inspect the extracted workload records independently from the statistical models.

Table 1. Dataset components and experimental uses.

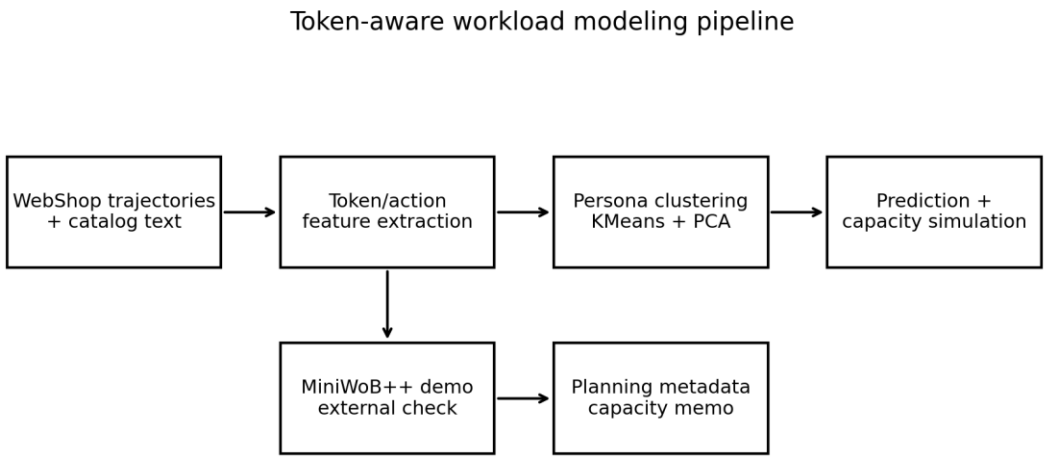
Component	Records	Unit	Use
WebShop human trajectories	1643	jsonl trajectories	workload extraction, clustering, token prediction, high-cost detection, capacity simulation
WebShop product catalog subset	1000	products	catalog text statistics and feature calibration
WebShop human instructions	12251	crowd instructions	instruction distribution and attribute checks
WebShop instruction attributes	1000	product attribute entries	schema consistency checks
MiniWoB++ demos	1467	demos sampled across 60 task directories	external validation of action/token workload patterns

Table 2. Feature schema used to convert browser trajectories into workload units.

Feature	Definition	Stage
instruction_tokens	Lexical token count in the natural-language shopping instruction	planning-time
first_query_tokens	Token count in the first human search query observed in the trajectory	early trajectory
attribute_count	Number of explicit semantic attributes in the goal record	planning-time
goal_option_count	Number of explicit option constraints such as color or size	planning-time

Feature	Definition	Stage
product_text_tokens	Tokens in goal product name, category path, query, attributes, and options	planning-time
trajectory_steps	Observed page transitions before the terminal state	trajectory
candidate_action_count	Sum of visible search-result, navigation, item, option, and buy candidates	trajectory
total_prompt_tokens	Sum of instruction, observation, and action-schema token proxies across steps	trajectory target
backtrack_count	Transitions back to result pages, item pages, or the start page	trajectory
success	Terminal reward equal to 1.0	outcome

Fig. 1. Pipeline for converting browser interaction data into token-aware workload personas and capacity inputs.



Results and Discussion

The WebShop categories are balanced enough to support cross-category comparison. Beauty, electronics, fashion, garden, and grocery each contribute between 289 and 374 trajectories. The average instruction length is narrow, ranging from 23.7 tokens in beauty to 25.9 tokens in fashion, so differences in serving demand are driven more by interaction behavior than by instruction length alone. Fashion tasks have the largest mean trajectory length, 12.0 steps, and the largest mean prompt-token proxy, 1,490.5. Grocery has the highest success rate at 53.6%, while fashion has a lower success rate of 44.2%. This pattern suggests that workload and reward are related but not identical: a category can

be costly because users inspect more products or options, even if the instruction is not longer.

The category table also shows that category is a weak but useful planning signal. Electronics and garden have similar mean step counts, 9.99 and 9.89, but electronics has a slightly higher mean prompt-token proxy. Beauty has the shortest mean instruction length but still produces substantial candidate exposure because cosmetic and personal-care products often contain option lists, ingredient descriptions, and small product differences. Grocery has the highest success rate despite a mean token proxy above 1,100, suggesting that strong product cues can offset moderate workload. These observations justify including category indicators in the predictive models while also showing why category alone cannot determine routing.

The token distribution is heavy-tailed. The median WebShop trajectory uses 722 prompt-token units, but the mean is 1,239 and the maximum reaches 13,893. This heavy tail is visible in Fig. 2 and explains why capacity planning based only on mean demand would understate risk. A small number of trajectories can occupy a large share of context and action-processing capacity. The same tail also explains why a simple planning-time regressor struggles: many long trajectories are caused by search depth, comparison, or backtracking that is only weakly signaled by the initial instruction.

KMeans model selection shows that k=2 has the highest silhouette, 0.603, because it cleanly separates direct buyers from all longer paths. For operational planning, k=4 is more useful: it preserves a strong silhouette of 0.302 while separating three distinct non-direct workloads. The four personas in Table 5 are not merely statistical clusters; they correspond to interpretable serving lanes. Direct buyers dominate the sample with 73.5% share and only 5.58 steps on average. Option comparators cover 19.2% of traces and require 1,992 token units. Recovery browsers cover 6.15% and average 4,545 token units. Catalog scanners are rare, 1.16%, but average 9,499 token units and 73 actions. A single rare persona therefore has an outsized effect on tail latency.

The persona scatter in Fig. 3 confirms that the Direct buyer cluster is compact, while long-horizon browsing occupies stretched regions of the PCA plane. Catalog scanners are far from the main mass because they combine many search pages, many item pages, high candidate counts, and repeated backtracking. Recovery browsers sit between option comparison and catalog scanning, with high backtracking and detail-page counts but less extreme search depth. This separation supports a capacity policy in which Direct buyers can use a standard low-latency lane, while Recovery browsers and Catalog scanners are routed to a conservative batching lane with larger context headroom [28].

Planning-time token regression is intentionally difficult because the predictors exclude post-hoc action counts. Ridge regression performs best, with MAE 853.8 and RMSE 1,328.7 token units. The mean baseline has RMSE 1,344.0, while Random Forest regression reaches 1,349.8. The low R2 values show that the observed trajectory length is driven by behavioral contingencies not fully visible from the instruction and product fields. Table 7 reinforces this point. Adding constraint fields and product text provides only small improvements over instruction length alone. The first human query helps MAE, but not enough to make token demand highly predictable. The interpretation is operational: static metadata can triage the highest-risk tasks, but it cannot replace live monitoring of step count and backtracking.

The ablation results are especially important because they prevent overclaiming. Product text and first-query information do improve the feature representation, but they do not remove the long-tail error. This means that workload management should be staged. The first stage can use instruction and catalog metadata to identify obvious high-risk requests before they enter the main lane. The second stage should use live telemetry after each action: current prompt size, consecutive searches, repeated item visits, and backtracks [31], [32]. The experiments quantify the first stage and show why the second stage remains necessary for precise capacity control.

The high-cost detector gives a more useful pre-execution signal than direct regression. High cost is

defined as the top quartile of WebShop prompt-token proxy, with threshold 1,354.5. Logistic regression achieves AUC 0.688 and F1 0.503, beating the majority baseline and the Random Forest classifier in AUC. Precision is 0.421 and recall is 0.626, which is appropriate for a routing use case where missing high-cost tasks is more costly than occasionally sending a normal task to a larger lane. The model is not a publication-grade predictor of exact latency, but it is strong enough to support a first-stage high-cost flag.

Figure 4 explains why static metadata has limited but nonzero value. The strongest Random Forest predictors are first-query tokens, product-text tokens, and instruction tokens, followed by categorical and constraint fields. These variables capture the initial semantic size of the request, but they do not capture whether the user will have to inspect many near-miss products. Figure 5 shows the same phenomenon from another angle: most predictions lie near the center of the distribution, while the highest-cost trajectories remain hard to anticipate. This is a useful negative result because it warns against relying on initial prompts alone for admission control.

The persona profiles also show that candidate-action count is a stronger operational signal than raw step count alone. Direct buyers average 34.5 candidate actions across the trajectory; Option comparators average 117.3; Recovery browsers average 276.1; Catalog scanners average 627.6. Candidate exposure matters because a pointer-style or text-action agent must consider or encode the available choices at each step. Two trajectories with the same number of page transitions can therefore have different serving cost if one repeatedly opens dense result pages and the other follows a narrow option path.

Reward differences across personas are consistent with the workload interpretation. Direct buyers have the highest success rate, 52.4%, and mean reward 0.778. Catalog scanners and Recovery browsers have lower success rates, 21.1% and 31.7%, even though they spend far more actions and tokens. The long paths often represent difficult retrieval or recovery situations rather than simply careful browsing. This makes them important for both agent evaluation and infrastructure planning: they are expensive and less reliable, so they are natural targets for better search planning, early stopping, query reformulation, or human handoff [36], [37].

The capacity simulation converts these statistical differences into operational pressure. At 40 tasks per minute, utilization is 0.540 and only 6.16% of tasks wait more than two seconds. At 50 tasks per minute, utilization rises to 0.680 and wait-over-two-second risk reaches 17.7%. At 60 tasks per minute, utilization is 0.807, p95 wait is 11.3 seconds, and 38.1% of tasks wait more than two seconds. At 70 tasks per minute, the system approaches saturation, with utilization 0.970 and 85.5% of tasks waiting more than two seconds. The sharp transition is caused by the heavy-tailed persona mix: the average workload is manageable, but rare long tasks consume enough service time to destabilize the queue near saturation.

MiniWoB++ validation shows that action/token heterogeneity is not unique to shopping. The task-stratified sample covers click/select, mixed, form/text, and spatial/navigation families. Mixed tasks have the highest mean workload-token proxy, 138,526, while form/text tasks have the highest high-cost share, 36.3%, and the largest mean action count, 49.9. The absolute token proxies differ from WebShop because MiniWoB++ stores richer DOM traces, but the qualitative pattern is consistent: short instructions can still create high workload when the browser state is dense or the action sequence is long.

The validation sample is deliberately not used to tune WebShop thresholds. Its role is to test whether the measurement vocabulary transfers: instruction size, action count, state count, and state-text proxy should still reveal heterogeneity when the domain changes. The result is positive. The dominant MiniWoB++ source of cost is not product comparison but UI-state density and action sequence length. This difference is useful because it shows that the workload framework is not tied to a particular shopping page taxonomy. It can describe both catalog-driven browsing and small web interfaces whose DOM traces are verbose relative to their natural-language instructions.

The planning-metadata examples in Table 12 illustrate how the empirical personas can be turned

into capacity memos. The memo is deliberately small: it names the persona, records instruction tokens, action count, prompt-token units, risk level, and routing lane. This metadata is not a new agent policy. It is a serving-side annotation that can accompany a task in a queue, scheduler, or monitoring dashboard [33], [39]. The examples show why persona-level information is easier to interpret than raw token totals alone: a 9,502-token Catalog scanner indicates a search-depth problem, while a 4,396-token Recovery browser indicates a backtracking problem.

The product catalog features show why catalog text should be considered even when the main target is trajectory cost. In the 1,000-product subset, product text varies from short listings to very long descriptions and option-heavy pages. A browser agent that places product titles, descriptions, options, and reviews into a prompt can therefore experience token growth before it takes many actions. The trajectory results combine this product-text effect with action-count effects, which is why product_text_tokens is included in both regression and clustering features.

The MiniWoB++ validation sample also emphasizes that DOM density and task family shape workload. Click/select tasks are numerous and often short, whereas form/text and mixed tasks can involve many state updates and larger serialized page observations. This result supports the broader method even though the paper's claims remain centered on shopping. A capacity planner can apply the same extraction pattern to any web-agent log that records instructions, page states, and actions: convert states to token proxies, derive action counts, cluster workload shapes, then simulate arrival rates.

The capacity memos are designed to be human-readable because operational routing decisions often need to be audited [38], [40]. A scheduler might use only numerical thresholds, but engineers and researchers benefit from seeing why a task was classified as high risk. A memo that says Catalog scanner is more informative than a memo that reports only 9,502 token units. It points to search-depth pressure and suggests interventions such as query reformulation, result filtering, or a maximum-results policy.

The main conclusion from the results is that success and capacity are orthogonal enough to require separate measurement. Direct buyers have the highest success rate and the smallest token footprint. Catalog scanners and Recovery browsers have lower success and much higher token demand, but even successful long trajectories can be expensive. A browser-agent benchmark therefore becomes more useful for production planning when each trajectory is evaluated along two axes: task quality and serving cost. The proposed workflow adds this second axis without changing the underlying benchmark.

Table 3. WebShop category-level workload and outcome statistics.

Category	N	Mean inst. tokens	Mean steps	Median steps	Mean prompt tokens	Median prompt tokens	Mean candidates	Success rate	Mean reward
beauty	374	23.70	9.620	6.000	1,118.6	695.0	65.91	0.497	0.757
electronics	289	24.46	9.986	6.000	1,213.1	674.0	68.84	0.450	0.723
fashion	344	25.93	12.00	7.000	1,490.5	888.0	87.08	0.442	0.742
garden	332	24.34	9.895	6.000	1,198.8	701.5	68.84	0.440	0.741
grocery	304	24.58	9.556	5.000	1,172.1	646.5	69.23	0.536	0.755

Fig. 2. Distribution of WebShop prompt-token proxy per trajectory.

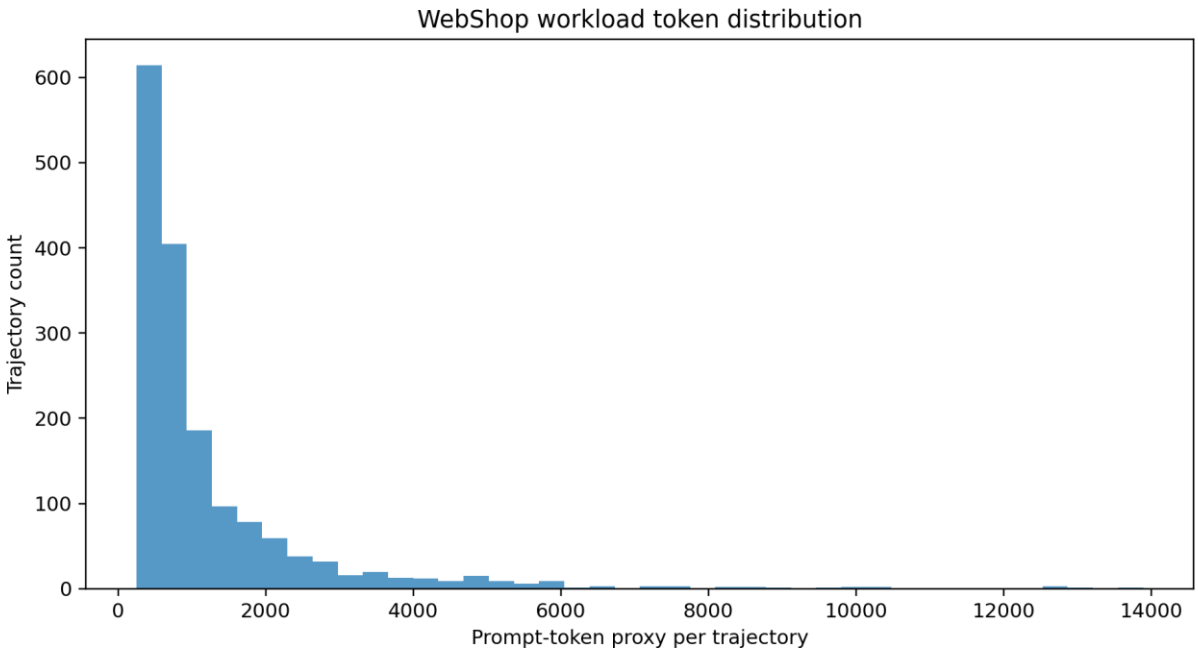


Table 4. KMeans model selection over standardized workload features.

k	Silhouette	Inertia
2.000	0.603	12,529.5
3.000	0.369	10,550.3
4.000	0.302	9,551.8
5.000	0.184	8,618.1
6.000	0.180	7,903.5
7.000	0.171	7,391.4
8.000	0.178	6,977.9

Table 5. Workload persona profiles from k=4 KMeans clustering.

Persona	N	Share	Mean tokens	Median tokens	Mean steps	Mean candidates	Search pages	Item pages	Detail pages	Backtracks	Success rate	Mean reward
Direct buyers	1208	0.735	636.7	582.5	5.575	34.48	1.444	2.638	0.336	0.692	0.524	0.778
Option	315	0.192	1,991.6	1,846.0	16.09	117.3	5.832	6.854	1.359	4.435	0.343	0.662

Persona	N	Share	Mean tokens	Median tokens	Mean steps	Mean candidates	Search pages	Item pages	Detail pages	Backtracks	Success rate	Mean reward
comparators												
Catalog scanners	19	0.012	9,499.0	9,502.0	73.00	627.6	37.11	25.58	4.053	23.05	0.211	0.661
Recovery browsers	101	0.061	4,544.8	4,396.0	35.76	276.1	15.11	13.29	3.901	11.84	0.317	0.612

Fig. 3. Persona clusters projected with PCA.

Table 6. Held-out token-demand prediction results.

Model	MAE	RMSE	R^2
Ridge	853.8	1,328.7	0.023

Model	MAE	RMSE	R^2
Mean baseline	881.4	1,344.0	-0.000
Random Forest	870.3	1,349.8	-0.009

Table 7. Ridge regression feature-group ablation.

Feature group	MAE	RMSE	R^2
instruction only	862.3	1,326.6	0.026
instruction + constraints	861.7	1,324.2	0.029
+ product text	861.7	1,324.2	0.029
+ first query	851.8	1,327.6	0.024
+ category	853.8	1,328.7	0.023

Fig. 4. Random Forest feature importances for planning-time token-demand prediction.

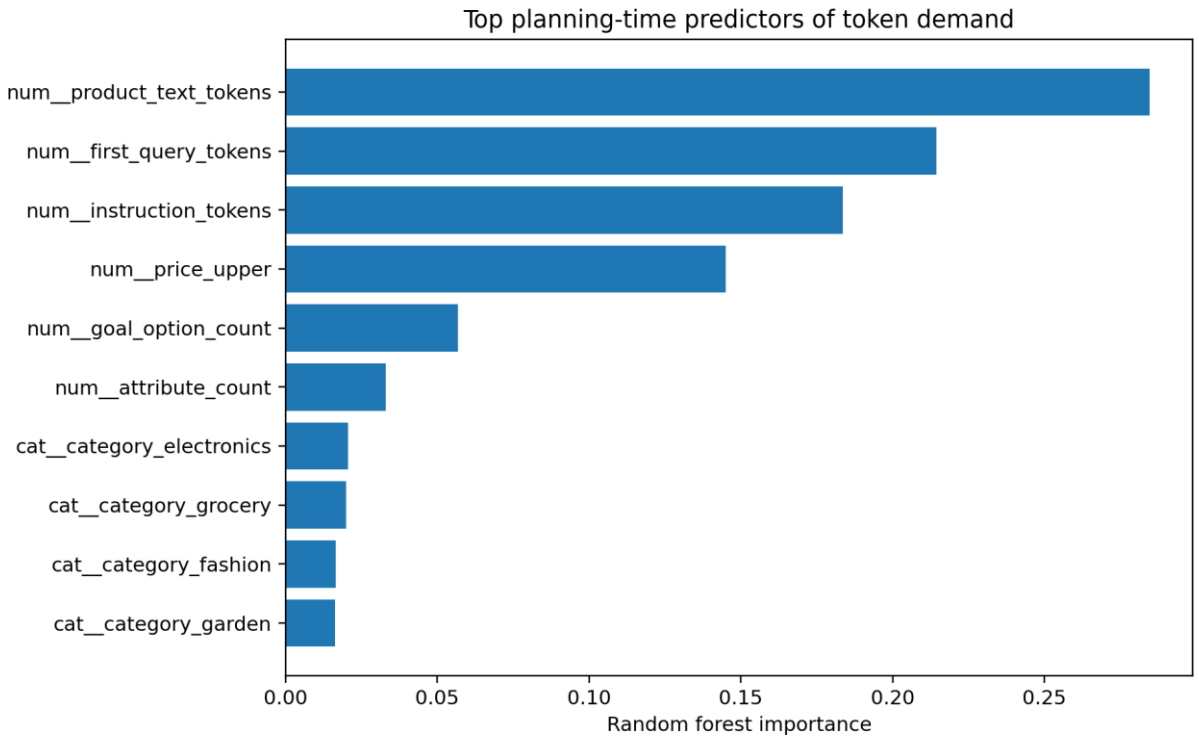


Fig. 5. Predicted versus observed prompt-token proxy for the best held-out regressor.

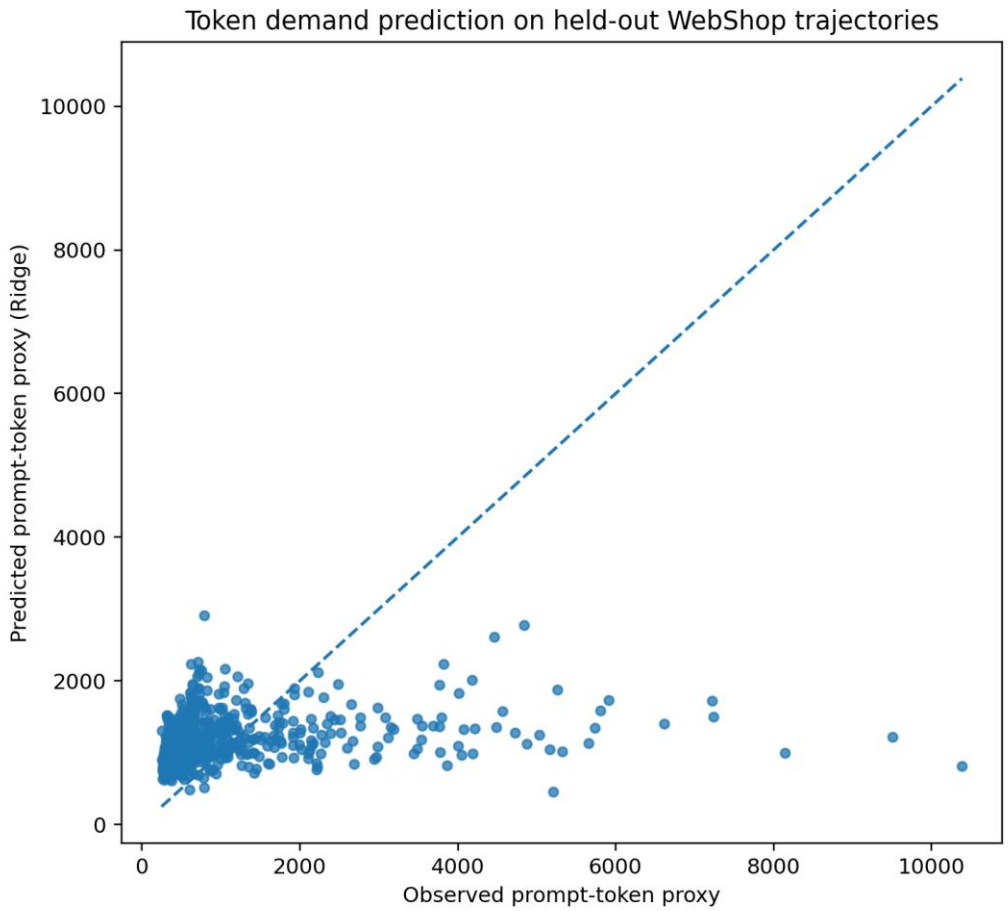


Table 8. High-cost task detection results using the top quartile of prompt-token proxy as the positive class.

Model	AUC	F1	Precision	Recall	Accuracy
Logistic Regression	0.688	0.503	0.421	0.626	0.692
Random Forest	0.636	0.335	0.425	0.276	0.726
Majority baseline	0.500	0.000	0.000	0.000	0.751

Table 9. Persona-level inputs used by the capacity simulation.

Persona	Share	Mean tokens	SD tokens	Mean steps
Catalog scanners	0.012	9,499.0	2,573.5	73.00
Direct buyers	0.735	636.7	271.5	5.575
Option comparators	0.192	1,991.6	613.8	16.09
Recovery browsers	0.061	4,544.8	1,204.4	35.76

Table 10. Capacity simulation summary for a four-worker serving lane.

Arrival/min	Servers	Jobs	Mean wait s	P95 wait s	Wait>2s rate	Utilization
10.00	4.000	1,792.0	0.001	0.000	0.000	0.130
20.00	4.000	3,493.0	0.031	0.000	0.004	0.271
30.00	4.000	5,391.0	0.166	1.075	0.027	0.417
40.00	4.000	7,175.0	0.366	2.506	0.062	0.540
50.00	4.000	8,977.0	1.031	5.662	0.177	0.680
60.00	4.000	10,591.0	2.719	11.33	0.381	0.807
70.00	4.000	12,630.0	18.14	47.24	0.855	0.970

Fig. 6. Under-provisioning and utilization curves under the observed persona mix.

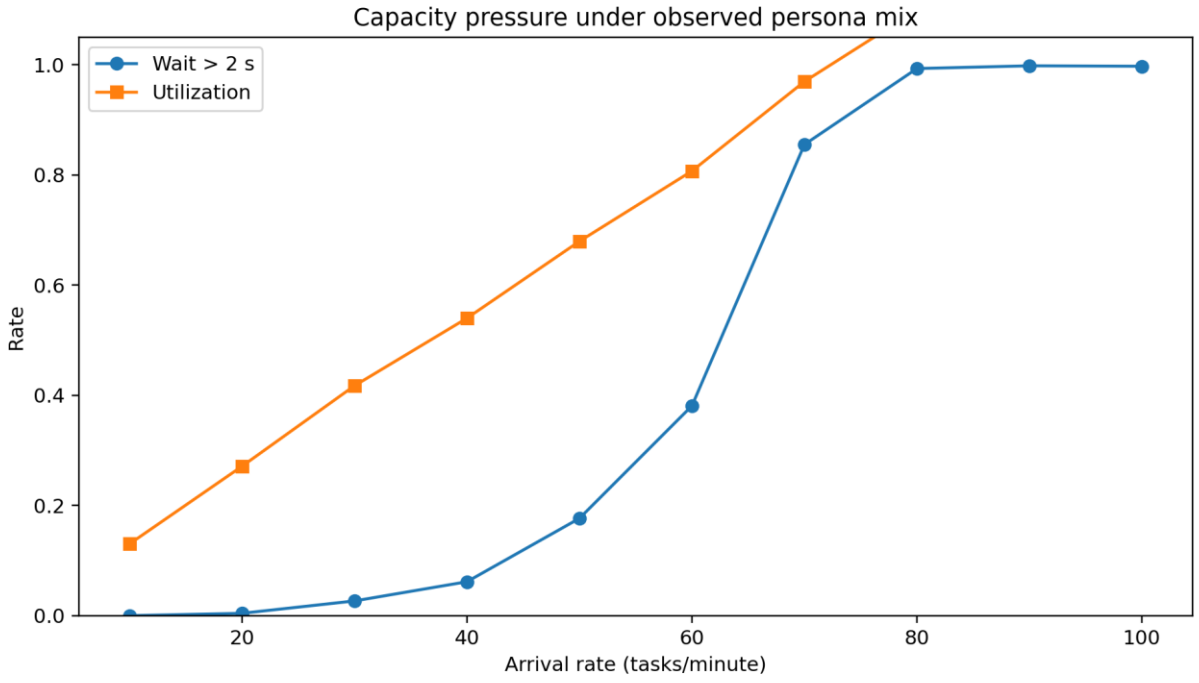


Table 11. MiniWoB++ task-family validation statistics from the deterministic task-stratified demo sample.

Task family	Demos	Tasks	Mean actions	Median actions	Mean workload tokens	High-cost share	Success rate
click/select	805	33	18.28	12.00	41,112.7	0.194	0.087

Task family	Demos	Tasks	Mean actions	Median actions	Mean workload tokens	High-cost share	Success rate
mixed	350	14	46.86	17.00	138,525.6	0.311	0.063
form/text	262	11	49.88	54.00	44,650.1	0.363	0.000
spatial/navigation	50	2	44.60	40.00	32,616.9	0.140	0.000

Table 12. Capacity-metadata memos generated for representative trajectories.

Persona	Trajectory	Category	Instruction excerpt	Capacity memo
Catalog scanners	20220513_2_fixed_200_392	fashion	i need some steel toed shoes that are chocolate colored and are a size 7, and price lower th...	Catalog scanners: instruction 24 tokens, 79 actions, 9502 prompt-token units; capacity risk high; route to reserve larger context and batch conservatively.
Direct buyers	20220504_1_113	beauty	i want to buy a brush for facial cleansing which is suitable for sensitive skin and it's blu...	Direct buyers: instruction 30 tokens, 5 actions, 582 prompt-token units; capacity risk normal; route to standard low-latency lane.
Option comparators	20220504_0_qual_153	fashion	i'm looking for a wide leg jeans with regular fit and tummy control. also, choose 3x large z...	Option comparators: instruction 36 tokens, 14 actions, 1846 prompt-token units; capacity risk high; route to reserve larger context and batch conservatively.
Recovery browsers	20220504_1_73	fashion	i am interested in buying a x-small size purple colored classic fit birthday t-shirt, and pr...	Recovery browsers: instruction 27 tokens, 31 actions, 4396 prompt-token units; capacity risk

Persona	Trajectory	Category	Instruction excerpt	Capacity memo
				high; route to reserve larger context and batch conservatively.

Limitations

The analysis is limited by the information present in the trajectory files. WebShop trajectories record page states and terminal rewards, but they do not include GPU traces, model-specific tokenization, actual wall-clock latency, or explicit human click timestamps. The total prompt-token proxy therefore measures relative serving pressure rather than exact latency. It is still useful for comparing trajectories because it is computed consistently across the corpus, but deployment engineers would recalibrate the service-time formula with logs from their own model and hardware.

The planning-time predictors intentionally omit post-hoc action counts. This makes the regression task realistic for early routing, but it also limits achievable accuracy. Once an agent has already taken several actions, live features such as current step count, repeated searches, and backtracks would improve cost estimates. The present design treats those live features as trajectory outcomes, not as pre-execution predictors.

The k=4 persona taxonomy is operational rather than universal. A different shopping site, search index, agent policy, or product catalog could change the cluster boundaries. The k-selection table shows that k=2 has the strongest silhouette, while k=4 offers more actionable separation of long-horizon workloads. This choice is appropriate for capacity planning, but it should be revisited when logs come from a different environment.

MiniWoB++ is used only as an external validation sample. Its DOM-rich traces are not directly comparable to WebShop product-search pages, and its reward values reflect the demonstration files rather than a newly trained agent. The validation result should be read as evidence that web-interaction workloads are heterogeneous across task families, not as a direct benchmark comparison between WebShop and MiniWoB++.

Another limitation is that the candidate-action proxy is deliberately conservative. It counts visible options and navigation opportunities from the recorded page structure, but it does not model the probability that an agent will encode every candidate in full. Different prompting templates may compress candidates, rank them, or use retrieval to include only a subset. The relative differences among personas should remain informative, but absolute candidate-token costs will depend on the prompting design.

The capacity simulation uses independent arrivals and stationary persona probabilities. Real traffic can be bursty, correlated with time of day, or shifted by product launches and seasonal shopping. A production system should therefore rerun the same simulation with observed arrival traces and time-varying persona mixtures. The simple Poisson model is still useful as a first controlled stress test because it makes the relationship between persona mix, service demand, and saturation easy to interpret.

The experiments also treat the lexical token proxy as a stable measurement layer. This choice is appropriate for comparing workloads within the same corpus, but different model tokenizers and prompt templates will change absolute token counts. A deployment that inserts screenshots, tool traces, chain summaries, or retrieval snippets would need to recompute the workload features under that template [36], [39]. The contribution is the measurement pipeline and the observed relative structure, not a universal conversion from page text to model cost.

Finally, the study focuses on browser shopping agents. The title and claims do not extend to all

browser agents, enterprise web automation, or multi-site browsing. The method can be adapted to those settings, but the empirical personas and thresholds in this paper are specific to the shopping and web-interaction datasets analyzed here.

Conclusion

This paper reframes browser shopping trajectories as LLM serving workload units. Using WebShop human trajectories and product/instruction data, the experiments show that prompt-token demand is heavy-tailed, that direct shopping tasks dominate frequency but not tail risk, and that rare catalog-scanning and recovery-heavy trajectories create disproportionate capacity pressure. The four-persona taxonomy gives a compact operational vocabulary for routing and monitoring browser-agent tasks.

The predictive experiments show that exact token demand is hard to infer from initial metadata alone, but high-cost detection is feasible enough to support triage. The capacity simulation then shows why such triage matters: under the observed persona mix, the four-worker lane remains comfortable through moderate arrivals but rapidly degrades near saturation as long trajectories accumulate. MiniWoB++ validation confirms that action/token heterogeneity also appears in non-shopping web tasks.

The experiments also show that workload modeling can be useful before a new agent is trained. Human trajectories provide enough variation to expose common workload shapes, and demonstration traces from a different web benchmark reproduce the broader action/token pattern. This makes the method suitable as an early design tool: before deploying a browser agent at scale, engineers can run the extractor on existing traces, estimate tail demand, and choose routing thresholds that match their service objectives.

The practical implication is straightforward. Browser-agent evaluation should report not only reward and success, but also token, action, and capacity-pressure measures. These measures help designers decide which tasks need larger contexts, conservative batching, or special routing. A lightweight persona memo can make those decisions transparent without changing the agent itself.

References

[1] S. Yao, H. Chen, J. Yang, and K. Narasimhan, "WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents," in Proc. Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 20744-20757.

[2] T. Shi, A. Karpathy, L. Fan, J. Hernandez, and P. Liang, "World of Bits: An Open-Domain Platform for Web-Based Agents," in Proc. International Conference on Machine Learning, 2017, pp. 3135-3144.

[3] E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang, "Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration," in Proc. International Conference on Learning Representations, 2018.

[4] R. Nakano et al., "WebGPT: Browser-Assisted Question-Answering with Human Feedback," arXiv:2112.09332, 2021.

[5] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," arXiv:2210.03629, 2022.

[6] T. B. Brown et al., "Language Models are Few-Shot Learners," in Proc. Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1877-1901.

[7] A. Chowdhery et al., "PaLM: Scaling Language Modeling with Pathways," arXiv:2204.02311, 2022.

[8] S. Zhang et al., "OPT: Open Pre-trained Transformer Language Models," arXiv:2205.01068, 2022.

[9] A. Vaswani et al., "Attention Is All You Need," in Proc. Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.

[10] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, "Orca: A Distributed Serving System for Transformer-Based Generative Models," in Proc. 16th USENIX Symposium on Operating Systems Design and Implementation, 2022, pp. 521-538.

[11] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in Proc. Fifth Berkeley Symposium on

Mathematical Statistics and Probability, 1967, pp. 281-297.

[12] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," Philosophical Magazine, vol. 2, no. 11, pp. 559-572, 1901.

[13] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," arXiv:1802.03426, 2018.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in Proc. KDD, 1996, pp. 226-231.

[15] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," Technometrics, vol. 12, no. 1, pp. 55-67, 1970.

[16] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[17] D. R. Cox, "The Regression Analysis of Binary Sequences," Journal of the Royal Statistical Society: Series B, vol. 20, no. 2, pp. 215-232, 1958.

[18] T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861-874, 2006.

[19] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation," Journal of Machine Learning Technologies, vol. 2, no. 1, pp. 37-63, 2011.

[20] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[21] W. McKinney, "Data Structures for Statistical Computing in Python," in Proc. 9th Python in Science Conference, 2010, pp. 56-61.

[22] C. R. Harris et al., "Array Programming with NumPy," Nature, vol. 585, pp. 357-362, 2020.

[23] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[24] L. Kleinrock, Queueing Systems, Volume 1: Theory. New York, NY, USA: Wiley, 1975.

[25] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, Quantitative System Performance: Computer System Analysis Using Queueing Network Models. Englewood Cliffs, NJ, USA: Prentice-Hall, 1984.

[26] Shenghan Lu and David Zhou, "LLM-Augmented Customer Representation Learning for Next-Purchase Prediction in Online Retail", JACS, vol. 3, no. 3, pp. 50-64, Mar. 2023, doi: 10.69987/JACS.2023.30305.

[27] Xiaohan Chang, Tong Ye, and Sophia Luo, "LLM-as-Reranker for Personalized Recommendation: Popularity Bias Mitigation and Faithful Natural-Language Explanations on MovieLens 100K", JACS, vol. 3, no. 8, pp. 61-78, Aug. 2023, doi: 10.69987/JACS.2023.30806.

[28] Shilu He, Haowei Tu, and Isa Liu, "Safe PD Capacity Forecasting with Time-Series Foundation Models and Calibrated Uncertainty for Heterogeneous GPU Clusters", JACS, vol. 3, no. 4, pp. 48-66, Apr. 2023, doi: 10.69987/JACS.2023.30404.

[29] Eric Wang and Heyu Wang, "QoE-Driven Reinforcement Learning for Joint Bitrate, Rebuffering, and TTFF Optimization in HLS/DASH", JACS, vol. 3, no. 2, pp. 50-59, Feb. 2023, doi: 10.69987/JACS.2023.30204.

[30] Yunhe Li, "Risk-Sensitive Offline Reinforcement Learning for Stable ABR QoE Improvements on Real HSDPA and LTE Traces", JACS, vol. 3, no. 4, pp. 1-11, Apr. 2023, doi: 10.69987/JACS.2023.30401.

[31] Ge Liu, Shilu He, and Isa Liu, "LLM-Augmented Multi-Source Root Cause Attribution for CPU and Network Faults in Microservices", JACS, vol. 3, no. 6, pp. 39-57, Jun. 2023, doi: 10.69987/JACS.2023.30604.

[32] Jiayi Nie and David Zheng, "Ambiguity-Aware HDFS Log Anomaly Detection with Retrieval-Augmented Failure Narratives and Selective Refusal", JACS, vol. 3, no. 1, pp. 66-80, Jan. 2023, doi: 10.69987/JACS.2023.30105.

[33] Kai Zhang, Siquan Meng, and Eric Zhou, "Evidence-Grounded Trading Desk Risk Memos over SEC Filings: Retrieval-Augmented Generation with XBRL Numeric Verification", JACS, vol. 3, no. 2, pp. 60-76, Feb. 2023, doi: 10.69987/JACS.2023.30205.

[34] Yuanzheng Chen, Yitian Zhang, David Chau, and Matt Sherman, "Credit Card Default Risk Tiering with Probability Calibration and Uncertainty-Driven Rejection: A Reproducible Study on the UCI Credit Card Clients Dataset", JACS, vol. 3, no. 4, pp. 31-47, Apr. 2023, doi: 10.69987/JACS.2023.30403.

[35] Ziliang Samuel Zhong, Ruiyan Ma, and Hailey Zhao, "Human-Uncertainty Distillation for Calibrated Vision Models on CIFAR-10H", JACS, vol. 3, no. 2, pp. 77-89, Feb. 2023, doi: 10.69987/JACS.2023.30206.

[36] Yunhe Li, "Execution-Feedback and Retrieval-Augmented Generation for Conversational Text-to-SQL: From One-Shot Questions to Clarification-Driven Executable Dialogs", JACS, vol. 3, no. 2, pp. 1-17, Feb. 2023, doi: 10.69987/JACS.2023.30201.

[37] Chenyu Li, Wenhao Su, and Eric Zhang, "Lightweight Hallucination Firewall for Enterprise LLM Applications: Evidence Consistency, Self-Checking, and Small-Model Detection on TruthfulQA", JACS, vol. 3, no. 1, pp. 49-65, Jan. 2023, doi: 10.69987/JACS.2023.30104.

[38] Daren Zheng, Chenyu Li, and Harvey Davidson, "Continual Red-Teaming for In-the-Wild Jailbreaks via Online Guardrail Updates and Guardrail Distillation", JACS, vol. 3, no. 2, pp. 35-49, Feb. 2023, doi: 10.69987/JACS.2023.30203.

[39] Qiyou Wu, Jingwen Bai, and Xiaohan Zhou, "Evidence-Grounded Financial RAG: Reducing Numerical Hallucination in LLM-Generated Corporate Risk Memos", JACS, vol. 3, no. 3, pp. 65-84, Mar. 2023, doi: 10.69987/JACS.2023.30306.

[40] Sihan Zhou, Zeyi Li, and Eric Wang, "Evidence-Grounded RAG for Tokenized Trade Receivable Disclosure QA under U.S. Capital Market Standards", JACS, vol. 3, no. 7, pp. 41-57, Jul. 2023, doi: 10.69987/JACS.2023.30704.