Language Technologies Institute

Serving the Agent Economy: Retrieval and Evaluation for Distributed Agents

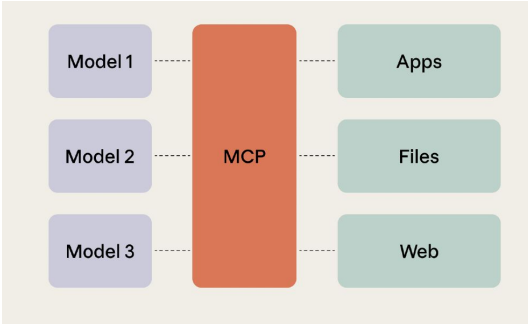
To Eun Kim

Carnegie Mellon University

Invited talk @ Sooth labs

Sep 17 2026

Distributed AI Agents



Distributed AI Agents forming ecosystems

- Era of experience
  - Procedural knowledge & experience data are valuable
- Agent artifacts are being shared
  - Trajectories, reasoning traces, skills, tools, documents
- What's missing:
  - A good platform where agents can **search and contribute** their artifacts.
  - Good **evaluation**, measuring "market-level" outcomes.

Google Research

WikiSkill: Compiling Agent Experience into Persistent Knowledge for Skill Evolution

SkillNet

Databricks Marketplace

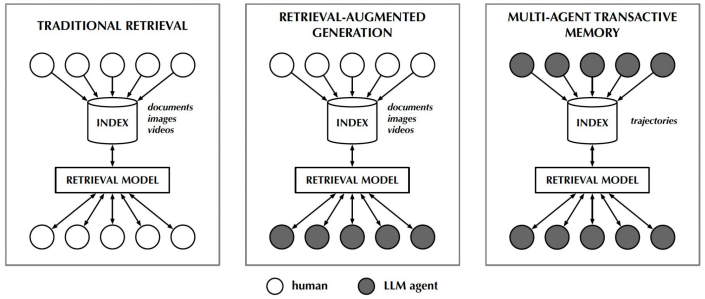
Open marketplace for data, analytics and AI

Discover AI Agents and Tools in AWS Marketplace

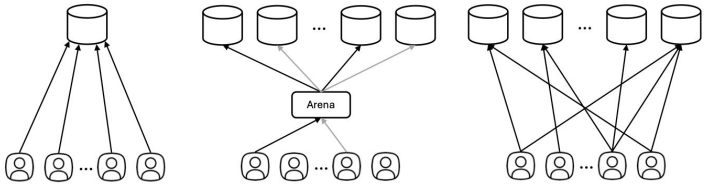
Explore thousands of agents, tools, and services from AWS Partners to deploy, customize, and scale AI agents on AWS, including on Amazon Bedrock AgentCore.

Questions for today

- How do we serve a population of agents?
  - Through a shared repository where agents can search and produce agent artifacts.
  - **Multi-Agent Transactive Memory** (preprint, 2026). ***T.E. Kim***, *X. He*, *D. Jain*, *A. Agrawal*, *N. Arabzadeh*, *F. Diaz*.



- How do we evaluate a market composed of distributed agents?
  - Agent-market simulation (e.g., market entry event forecasting)
  - **Evaluation of Agents under Simulated Marketplace Dynamics** (SIGIR 2026). ***T.E. Kim***, *A. Salemi*, *H. Zamani*, *F. Diaz*.



Language Technologies Institute

BAIR

Multi-Agent Transactive Memory

To Eun Kim
CMU

Xuhong He
CMU

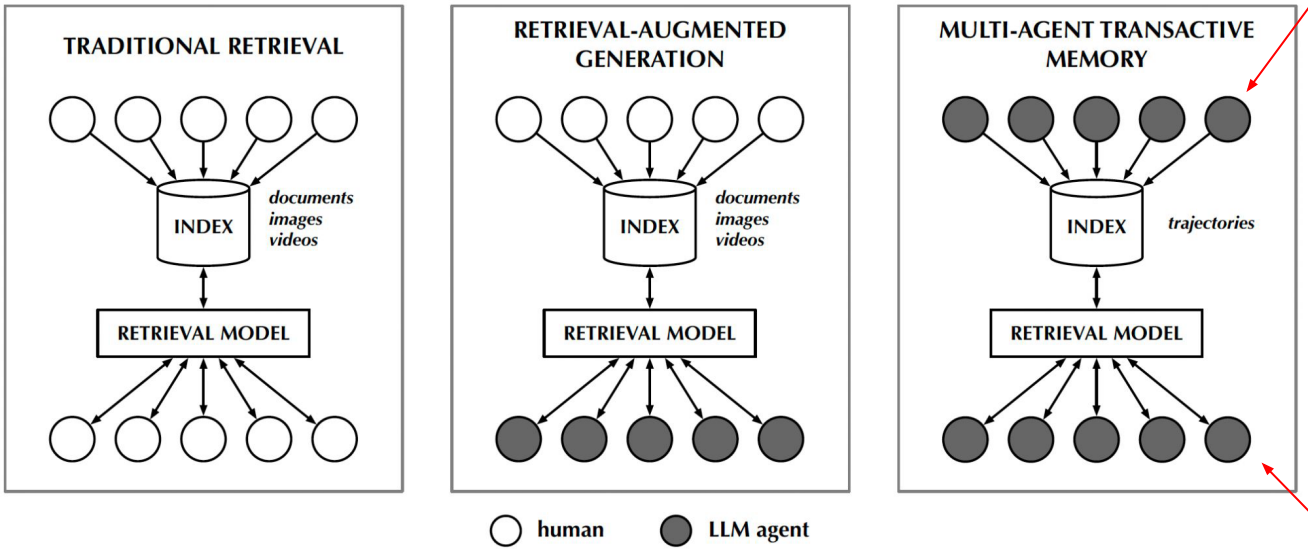
Dishank Jain
CMU

Ambuj Agrawal
CMU

Negar Arabzadeh
UC Berkeley

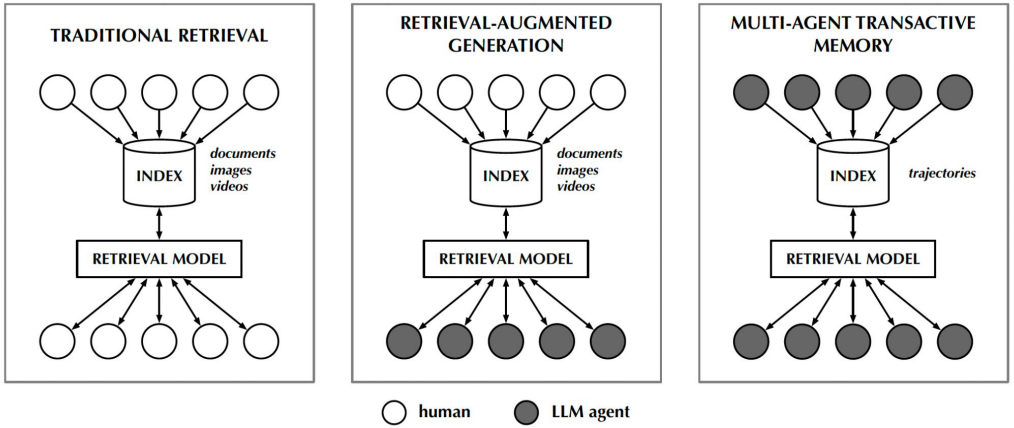
Fernando Diaz
CMU

MATM: Multi-Agent Transactive Memory



Research Questions

1. Given a set of agent users (consumers), does retrieving from population level memory helpful?
2. Can we learn to rank agent trajectories? (reranker training)
3. Is retrieval benefit distributed across all participants or concentrated to some agents (e.g., small models)?
4. Do we have to organize artifacts by task? Can agents also benefit from different task's trajectories?
5. Does the population memory scale?



System setup

- Agents' task / environment (interactive env)
  - ALFWorld: text-based household task environment
    - Various task and map variations
  - WebArena: web navigation-based task solving
    - Long and complex observations; relatively small action space

```
You are in the middle of a room. Looking quickly around you, you see a drawer 2, a
shelf 5, a drawer 1, a shelf 4, a sidetable 1, a drawer 5, a shelf 6, a shelf 1, a
shelf 9, a cabinet 2, a sofa 1, a cabinet 1, a shelf 3, a cabinet 3, a drawer 3, a
shelf 11, a shelf 2, a shelf 10, a dresser 1, a shelf 12, a garbagecan 1, a armchair
1, a cabinet 4, a shelf 7, a shelf 8, a safe 1, and a drawer 4.

Your task is to: put some vase in safe.

> go to shelf 6
You arrive at loc 4. On the shelf 6, you see a vase 2.

> take vase 2 from shelf 6
You pick up the vase 2 from the shelf 6.

> go to safe 1
You arrive at loc 3. The safe 1 is closed.

> open safe 1
You open the safe 1. The safe 1 is open. In it, you see a keychain 3.
```

ALFWorld

webarena.onestopshop.com

```
[4] RootWebArea 'Patio, Lawn ..'
  [1543] link 'Image'
    [1547] img 'Image'
  [1552] link 'Outdoor Patio..'
  [1549] LayoutTable ''
    [1559] StaticText 'Rating:'
    [1557] generic '82%'
    [1567] link '12 Reviews'
  [1574] StaticText '$49.99'
  [1582] button 'Add to Cart' focusable:
True
  [1585] button 'Wish List' focusable: ...
  [1586] button 'Compare' focusable: ...
```

WebArena (Observation)

Action Type	Description
`noop`	Do nothing
`click(elem)`	Click at an element
`hover(elem)`	Hover on an element
`type(elem, text)`	Type to an element
`press(key_comb)`	Press a key comb
`scroll(dir)`	Scroll up and down
`tab_focus(index)`	focus on *i*-th tab
`new_tab`	Open a new tab
`tab_close`	Close current tab
`go_back`	Visit the last URL
`go_forward`	Undo `go_back`
`goto(URL)`	Go to URL

WebArena (Actions)

System setup

- Agents' task / environment (interactive env)
  - ALFWorld: text-based household task environment
  - WebArena: web navigation-based task solving

- Producer and Consumer agent users (population setup)
  - ALFWorld: 34 consumers; 35 producers
  - WebArena: 34 consumers; 37 producers

  - In the experiment, we are going to track all the provenance (i.e., who produced and consumed what content)

Model	Producer	Consumer
trained-seq2seq	A	
openai/gpt-4-turbo	W	
openai/gpt-4-turbo-preview	W	
anthropic/claude-3.5-sonnet	W	
openai/gpt-oss-20b	AW	AW
openai/gpt-oss-120b	AW	AW
openai/gpt-5.4	AW	AW
openai/gpt-5.4-nano	AW	AW
anthropic/claude-3-haiku	AW	AW
anthropic/claude-sonnet-4.6	AW	AW
anthropic/claude-opus-4	AW	AW
anthropic/claude-opus-4.6	AW	AW
google/gemini-2.5-flash-lite	AW	AW
google/gemini-2.5-flash	AW	AW
google/gemini-2.5-pro	AW	AW
google/gemma-4-31b-it	AW	AW
meta-llama/llama-3.1-8b-instruct	AW	AW
meta-llama/llama-3.2-3b-instruct	AW	AW
meta-llama/llama-3.3-70b-instruct	AW	AW
meta-llama/llama-4-maverick	AW	AW
deepseek/deepseek-r1	AW	AW
deepseek/deepseek-r1-0528	AW	AW
deepseek/deepseek-chat-v3.1	AW	AW
deepseek/deepseek-v3.2	AW	AW
qwen/qwen3-32b	AW	AW
qwen/qwen3-235b-a22b	AW	AW
qwen/qwen3-max	AW	AW
qwen/qwen3.5-flash-02-23	AW	AW
x-ai/grok-3-mini	AW	AW
x-ai/grok-4-fast	AW	AW
x-ai/grok-4.1-fast	AW	AW
x-ai/grok-4.20	AW	AW
z-ai/glm-4.5	AW	AW
z-ai/glm-5	AW	AW
minimax/minimax-m1	AW	AW
minimax/minimax-m2	AW	AW
mistralai/ministral-14b-2512	AW	AW
mistralai/mistral-large-2512	AW	AW

Table 3: Producer and consumer agents across ALFWorld (A) and WebArena (W). AW indicates the model serves in that role for both environments.

System setup

- Agents' task / environment (interactive env)
  - ALFWorld: text-based household task environment
  - WebArena: web navigation-based task solving

- Producer and Consumer agent users (population setup)
  - ALFWorld: 34 consumers; 35 producers
  - WebArena: 34 consumers; 37 producers

- We store raw interaction trajectories with key-value structure
  → Interaction trajectory = **{ (goal, obs, action) }**
  - Key: current goal + obs + action (retrieval key)
  - Value: next 5 steps {(obs, action)} (guidance)
    - With good retrieval, agents are guided with possible future steps and see what to expect
    - We only index successful trajectories

System setup

- Agents' task / environment (interactive env)
  - ALFWorld: text-based household task environment
  - WebArena: web navigation-based task solving
- Producer and Consumer agent users (population setup)
  - ALFWorld: 34 consumers; 35 producers
  - WebArena: 34 consumers; 37 producers
- We store raw interaction trajectories with key-value structure
  → Interaction trajectory = **{ (goal, obs, action) }**
  - Key: current goal + obs + action (retrieval key)
  - Value: next 5 steps {(obs, action)} (guidance)
- Pre-populate the shared corpus with publicly available agent trajectories

Model	Producer	Consumer
trained-seq2seq	A	
openai/gpt-4-turbo	W	
openai/gpt-4-turbo-preview	W	
anthropic/claude-3.5-sonnet	W	
openai/gpt-oss-20b	AW	AW
openai/gpt-oss-120b	AW	AW
openai/gpt-5.4	AW	AW
openai/gpt-5.4-nano	AW	AW
anthropic/claude-3-haiku	AW	AW
anthropic/claude-sonnet-4.6	AW	AW
anthropic/claude-opus-4	AW	AW
anthropic/claude-opus-4.6	AW	AW
google/gemini-2.5-flash-lite	AW	AW
google/gemini-2.5-flash	AW	AW
google/gemini-2.5-pro	AW	AW
google/gemma-4-31b-it	AW	AW
meta-llama/llama-3.1-8b-instruct	AW	AW
meta-llama/llama-3.2-3b-instruct	AW	AW
meta-llama/llama-3.3-70b-instruct	AW	AW
meta-llama/llama-4-maverick	AW	AW
deepseek/deepseek-r1	AW	AW
deepseek/deepseek-r1-0528	AW	AW
deepseek/deepseek-chat-v3.1	AW	AW
deepseek/deepseek-v3.2	AW	AW
qwen/qwen3-32b	AW	AW
qwen/qwen3-235b-a22b	AW	AW
qwen/qwen3-max	AW	AW
qwen/qwen3.5-flash-02-23	AW	AW
x-ai/grok-3-mini	AW	AW
x-ai/grok-4-fast	AW	AW
x-ai/grok-4.1-fast	AW	AW
x-ai/grok-4.20	AW	AW
z-ai/glm-4.5	AW	AW
z-ai/glm-5	AW	AW
minimax/minimax-m1	AW	AW
minimax/minimax-m2	AW	AW
mistralai/ministral-14b-2512	AW	AW
mistralai/mistral-large-2512	AW	AW

Table 3: Producer and consumer agents across ALFWorld (A) and WebArena (W). AW indicates the model serves in that role for both environments.

System setup

- Agents' task / environment (interactive env)
  - ALFWorld: text-based household task environment
  - WebArena: web navigation-based task solving

- Producer and Consumer agent users (population setup)
  - ALFWorld: 34 consumers; 35 producers
  - WebArena: 34 consumers; 37 producers

- We store raw interaction trajectories with key-value structure
  → Interaction trajectory = **{ (goal, obs, action) }**
  - Key: current goal + obs + action (retrieval key)
  - Value: next 5 steps {(obs, action)} (guidance)

- Pre-populate the shared corpus with publicly available agent trajectories
  - Then let the rest of the agents produce and consume! Go!

We assume that all agents have incentive/agreement to index their trajectories to the shared corpus

Model	Producer	Consumer
trained-seq2seq	A	
openai/gpt-4-turbo	W	
openai/gpt-4-turbo-preview	W	
anthropic/claude-3.5-sonnet	W	
openai/gpt-oss-20b	AW	AW
openai/gpt-oss-120b	AW	AW
openai/gpt-5.4	AW	AW
openai/gpt-5.4-nano	AW	AW
anthropic/claude-3-haiku	AW	AW
anthropic/claude-sonnet-4.6	AW	AW
anthropic/claude-opus-4	AW	AW
anthropic/claude-opus-4.6	AW	AW
google/gemini-2.5-flash-lite	AW	AW
google/gemini-2.5-flash	AW	AW
google/gemini-2.5-pro	AW	AW
google/gemma-4-31b-it	AW	AW
meta-llama/llama-3.1-8b-instruct	AW	AW
meta-llama/llama-3.2-3b-instruct	AW	AW
meta-llama/llama-3.3-70b-instruct	AW	AW
meta-llama/llama-4-maverick	AW	AW
deepseek/deepseek-r1	AW	AW
deepseek/deepseek-r1-0528	AW	AW
deepseek/deepseek-chat-v3.1	AW	AW
deepseek/deepseek-v3.2	AW	AW
qwen/qwen3-32b	AW	AW
qwen/qwen3-235b-a22b	AW	AW
qwen/qwen3-max	AW	AW
qwen/qwen3.5-flash-02-23	AW	AW
x-ai/grok-3-mini	AW	AW
x-ai/grok-4-fast	AW	AW
x-ai/grok-4.1-fast	AW	AW
x-ai/grok-4.20	AW	AW
z-ai/glm-4.5	AW	AW
z-ai/glm-5	AW	AW
minimax/minimax-m1	AW	AW
minimax/minimax-m2	AW	AW
mistralai/ministral-14b-2512	AW	AW
mistralai/mistral-large-2512	AW	AW

Table 3: Producer and consumer agents across ALFWorld (A) and WebArena (W). AW indicates the model serves in that role for both environments.

Trajectory retrieval

- Per timestep, an agentic *retrieval planner* decides a binary decision → if we want to retrieve future guidance, given the current state and observation and its reasoning (ReAct)
- First-stage: Vector search over the key (E5-base)
- Second-stage: Learning to Rank Trajectories (**LTRT**)
  - Rerank top-20 → agents only consume top-1
  - 44 Features
    - Producer benchmark information → aiming for trust modeling effect
    - Consumer ID → aiming for personalization effect

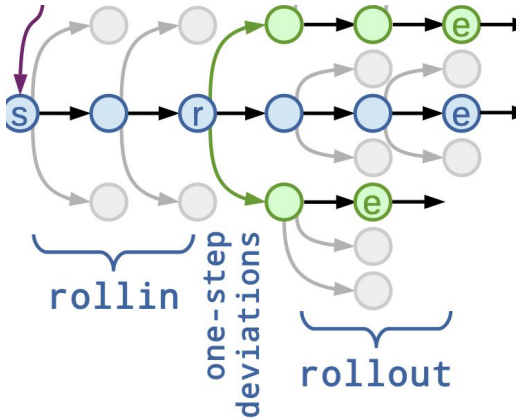
Category	Features (# features: 44)
Producer Agent Info (#: 13)	agent ID context-window agent benchmark scores (11 features): Artificial Analysis Intelligence (AAI) Index GDPval-AA τ^2-Bench Telecom Terminal-Bench Hard SciCode AA-LCR AA-Omniscience Accuracy IFBench Humanity's Last Exam (HLE) GPQA-Diamond CritPt
Consumer Agent Info (#: 1)	agent ID
1st Stage Retrieval (#: 1)	1st stage retrieval score
Query Features (#: 2)	query length current step number
Trajectory Features (#: 4)	retrieved chunk length number of steps in trajectory success flag trajectory length
Query–Trajectory Interaction Features (#: 23)	unigram text tfidf cosine similarity unigram goal tfidf cosine similarity unigram state tfidf cosine similarity unigram context tfidf cosine similarity bigram text tfidf cosine similarity bigram goal tfidf cosine similarity bigram state tfidf cosine similarity bigram context tfidf cosine similarity text overlap ratio goal overlap ratio state overlap ratio context overlap ratio text jaccard similarity goal jaccard similarity state jaccard similarity context jaccard similarity text embedding similarity goal embedding similarity state embedding similarity context embedding similarity task match task variation match step number difference

LTRT Features

LTRT training

Training Data Collection

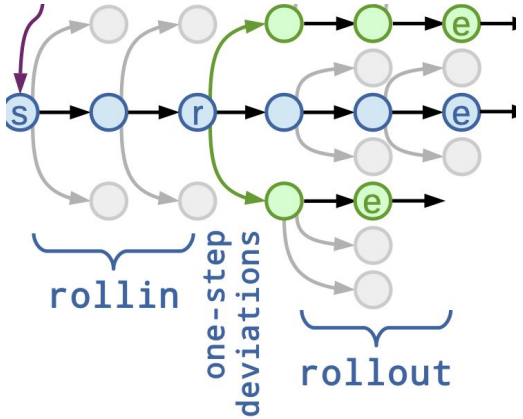
1. Baseline trajectory:
    1.1. full roll-out without retrieval
    1.2. Evaluate baseline trajectory score (s_base)
2. Retrieval-interrupted trajectories
    2.1. Roll-in to a random time-step and clone the state
    2.2. Retrieve from MATM
    2.3. Branch 5 roll-outs with 1st, 5th, 10th, 15th, and 20th chunks (Do agentic-RAG)
    2.4. For each roll-out,
        2.4.1. Score s_i
        2.4.2. Document relevance label → (s_i – s_base) (marginal utility-based labeling)
3. (Successful trajectories are added/indexed to MATM)



LTRT training

Training Data Collection

1. Baseline trajectory:
   1.1. full roll-out without retrieval
   1.2. Evaluate baseline trajectory score (s_base)
2. Retrieval-interrupted trajectories
   2.1. Roll-in to a random time-step and clone the state
   2.2. Retrieve from MATM
   2.3. Branch 5 roll-outs with 1st, 5th, 10th, 15th, and 20th chunks (Do agentic-RAG)
   2.4. For each roll-out,
      2.4.1. Score s_i
      2.4.2. Document relevance label → (s_i – s_base) (marginal utility-based labeling)
3. (Successful trajectories are added/indexed to MATM)



Models

- Lightweight Learning-To-Rank (LTR) models:
  - Pointwise FFN (CE loss)
  - Pairwise Lambdamart (ndcg ranking objective)
  - Pairwise SVMRank (pairwise hinge loss)

Metrics (consumer-side)

- Effectiveness: *Success Rate*
- Efficiency: *# interaction steps*
- Pareto-dominance: *Return-Paired Preference (break system ranking ties with efficiency)*

Findings

1. MATM-Augmentation improves effectiveness and efficiency
2. Learning to Rank Trajectories further improve MATM participants' average welfare
3. MATM benefits are distributed across the agent population
4. MATM offers cross-task generalization
5. MATM scales with memory size

1. MATM-Augmentation improves effectiveness and efficiency

Method	ALFWorld			WebArena		
	SR↑	**# steps↓**	**RPP↑**	**SR↑**	**# steps↓**	**RPP↑**
no retrieval	0.4708	11.7664	-0.1586	0.1818	21.9886	-0.0511
single-stage	0.5511	11.1796	-0.0451	**0.2045**	20.2614	0.0284
rerank-FFN	0.5861	11.0336	-0.0046	**0.2045**	**19.9091**	**0.0398**
rerank-LambdaMART	0.5715	10.8474	0.0608	0.1818	20.4205	-0.0341
rerank-SVMRank	**0.6431**	**10.3453**	**0.1474**	**0.2045**	20.2841	0.0170

Table 1: Evaluation of MATM-augmented agents in interactive environments. Success Rate (SR) and number of steps (# steps) are used for measuring the effectiveness and efficiency. Values are average of the five runs of randomized task-model allocation.

2. Learning to Rank Trajectories further improve MATM participants' welfare

Method	ALFWorld			WebArena		
	SR↑	**# steps↓**	**RPP↑**	**SR↑**	**# steps↓**	**RPP↑**
no retrieval	0.4708	11.7664	-0.1586	0.1818	21.9886	-0.0511
single-stage	0.5511	11.1796	-0.0451	**0.2045**	20.2614	0.0284
rerank-FFN	0.5861	11.0336	-0.0046	**0.2045**	**19.9091**	**0.0398**
rerank-LambdaMART	0.5715	10.8474	0.0608	0.1818	20.4205	-0.0341
rerank-SVMRank	**0.6431**	**10.3453**	**0.1474**	**0.2045**	20.2841	0.0170

Table 1: Evaluation of MATM-augmented agents in interactive environments. Success Rate (SR) and number of steps (# steps) are used for measuring the effectiveness and efficiency. Values are average of the five runs of randomized task-model allocation.

2. Learning to Rank Trajectories further improve MATM participants' welfare

Benchmark	Rank	Feature Name	Feature Category	Feature Importance
AlfWorld	1	SciCode score	Producer Agent Info	0.0596
AlfWorld	2	number of steps in trajectory	Trajectory Features	-0.0514
AlfWorld	3	AAI score	Producer Agent Info	0.0368
AlfWorld	4	HLE score	Producer Agent Info	-0.0233
AlfWorld	5	GPQA-Diamond score	Producer Agent Info	-0.0230
AlfWorld	6	state overlap ratio	Query–Trajectory Interaction	0.0188
AlfWorld	7	GDPval-AA score	Producer Agent Info	0.0185
AlfWorld	8	τ^2-Bench Telecom score	Producer Agent Info	-0.0168
AlfWorld	9	retrieved chunk length	Trajectory Features	0.0156
AlfWorld	10	IFBench score	Producer Agent Info	-0.0155
WebArena	1	AA-LCR score	Producer Agent Info	0.0069
WebArena	2	IFBench score	Producer Agent Info	0.0063
WebArena	3	number steps in trajectory	Trajectory Features	0.0044
WebArena	4	state embedding similarity	Query–Trajectory Interaction	0.0036
WebArena	5	goal overlap ratio	Query–Trajectory Interaction	0.0032
WebArena	6	GPQA-Diamond score	Producer Agent Info	0.0028
WebArena	7	retrieved chunk length	Trajectory Features	0.0024
WebArena	8	CritPt score	Producer Agent Info	0.0021
WebArena	9	current step number	Query Features	0.0020
WebArena	10	bigram text tfidf cosine similarity	Query–Trajectory Interaction	0.0019

Table 6: Top feature rankings for LTRT model on each benchmark. SVMRank for ALFWorld and FFN for WebArena. For SVMRank, we report the feature importance as the weight learnt for SVMRank. For FFN, we compute feature importance by removing the feature and measuring the drop in NDCG@10 score while training the LTRT reranker.

- Producer information features are important (producer trust modeling)

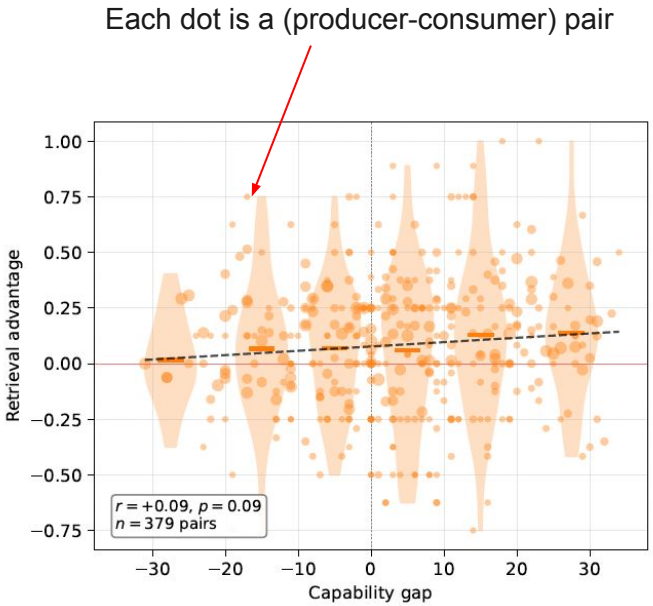
3. MATM benefits are distributed across the agent population

For each producer-consumer pairings, we see:

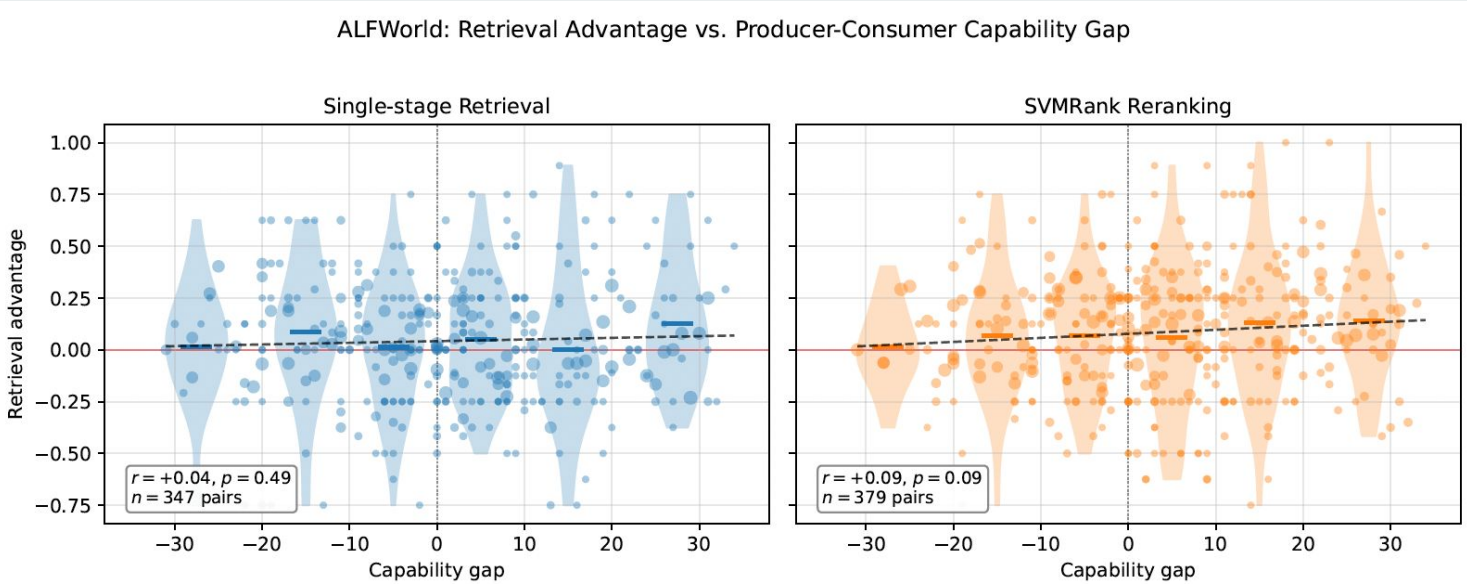
- **Retrieval advantage**
  - Gain in consumer success rate when retrieving from that producer relative to its no-retrieval baseline

- **Capability gap**
  - Capability = agent's aggregated benchmark scores (AAI index)
  - Capability gap = producer capability – consumer capability

  - Higher the gap, producer is more capable than the consumer



3. MATM benefits are distributed across the agent population



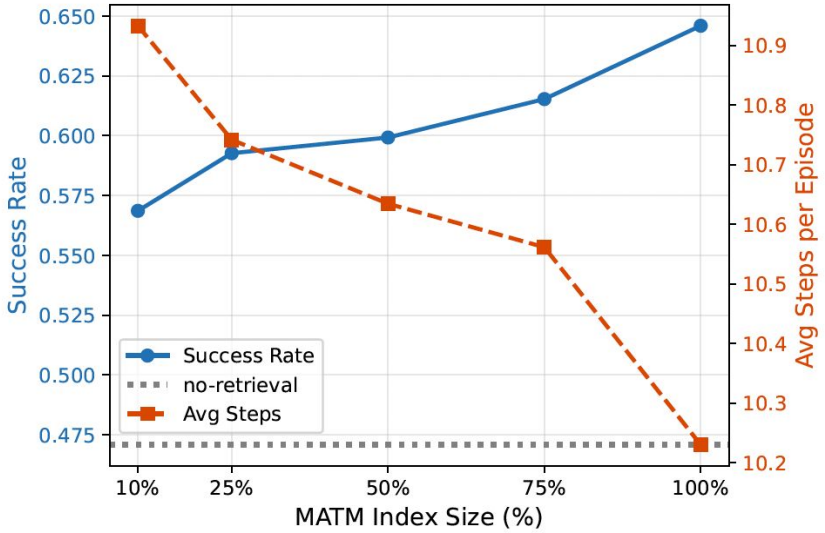
- Consumers benefit from retrieval regardless of whether the producer is weaker, comparable, or stronger than themselves
  → MATM system's value cannot be reduced to a few strong producers
- Slight upward trend: stronger producer may yield higher benefit to consumer (not significant)
- Reranking lifts the entire retrieval advantage distribution (confirms reranking benefit)

4. MATM offers cross-task generalization

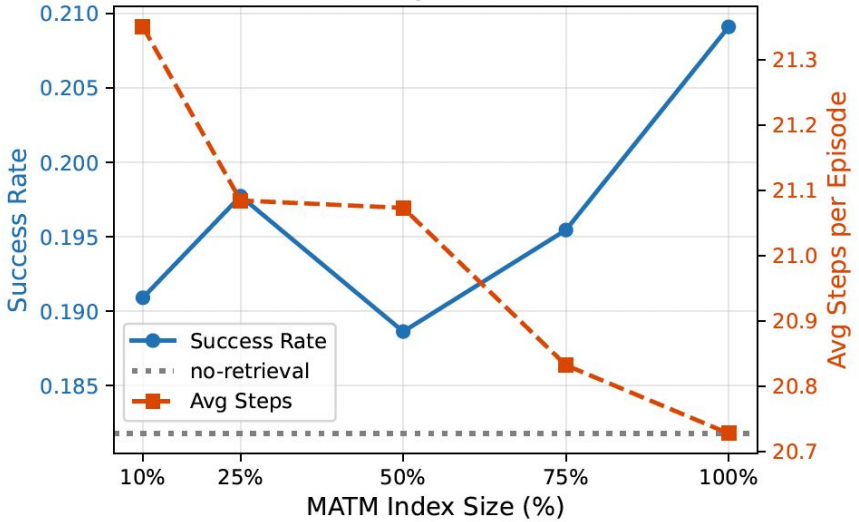
Retrieval Scope	SR ↑	# steps ↓	RPP ↑
ALFWorld (274)			
full	**0.6460**	**10.2300**	**0.0566**
same task type	0.6314	10.4635	0.0055
cross task type	0.5985	10.7445	-0.0620
WebArena (47)			
full	**0.2979**	19.0426	**0.1383**
same task type	0.2766	19.1915	0.0000
cross task type	0.1915	**19.0000**	-0.1383

- Same task > cross task; but:
- Full-retrieval achieves the highest performance
  → Restricting MATM to the same task type is not a good idea

5. MATM scales with memory size

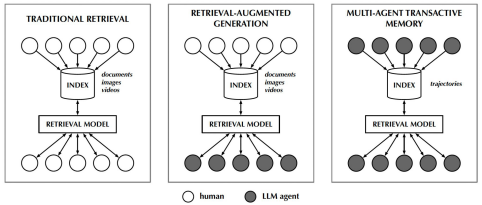


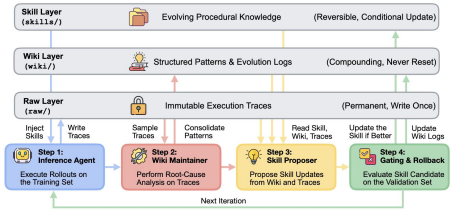
ALFWorld



WebArena

MATM Vs. WikiSkill





Agent population co-evolution	Individual evolution → skill transferability test
Artifacts = raw trajectories	Artifacts = skills (raw trajectories are stored but only for skill extraction)
Top-k Artifact Retrieval & Retrieval optimization	All skills fed into agent's prompt
Indexing structure key : value = context : next steps	Wiki-like content-based pattern extraction
No corpus maintaining	Explicit corpus maintaining (Wiki Maintainer)

Language Technologies Institute

SIGIR2026
MELBOURNE | NAARM
AUSTRALIA

Evaluation of Agents under Simulated AI Marketplace Dynamics

To Eun Kim
CMU

Alireza Salemi
UMass Amherst

Hamed Zamani
UMass Amherst

Fernando Diaz
CMU

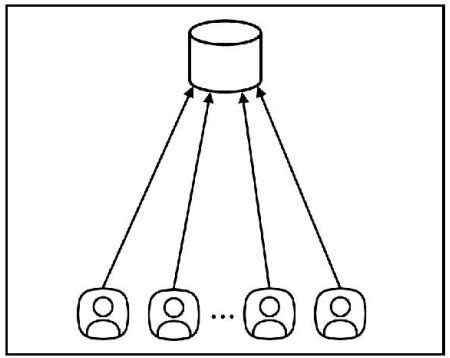
SIGIR 2026 Perspectives

Agents live in marketplaces

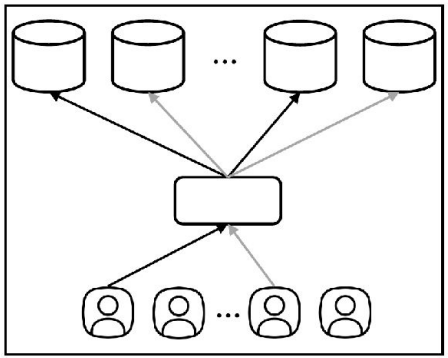
- Modern agents — LLMs, retrievers, tools — are orchestrated to deliver generated information objects through conversational interfaces.
- Agent–access is increasingly mediated by **marketplaces**.
- Platforms make it trivial for users and agents to try many providers, ***switching*** when a (perceived) better options appears.
- **Competition** between systems is therefore inherent to deployment.

Agents now live in marketplaces — distributed, discoverable, and competing to be selected.
Our evaluation should reflect that.

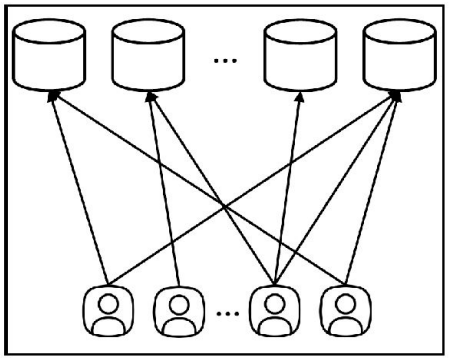
Evaluation Paradigms



Cranfield



Arena



Marketplace

new!

- **Static Benchmark (Cranfield)**: multi-user, single system.
  - All queries routed to a fixed system. *No competition, no user choice*.
- **Arena**: multi-user, multi-system.
  - A platform mediates anonymous pairwise comparisons. There is a certain-level of competition, but *users cannot form persistent preference or switching* (Arena platform decides).
- **Marketplace (ours)**: multi-user, multi-system.
  - *Users actively select* among competing providers, with no central mediator required — enabling competitive dynamics, behavioral adaptation, and market share over time.

Why competitive dynamics matter

- Network effects
  - a system's value depends on its adoption by the generator population and its differentiation from rivals, not just intrinsic quality.
- Winner-take-all
  - small performance differences can produce dramatic market-share disparities.
- Portfolio & path dependence
  - agents diversify across providers; early interactions alter future routing and resource allocation.
- Therefore, a deployment success is not a single offline number
  - it shows up as query traffic, market share, and sustained usage.

Motivating study (Cranfield model)

- Seven systems
- 500 factoid questions from SimpleQA; agents produce generated information objects (not short phrases).
- Score each answer for correctness and rank by F1.

Model	F1
Qwen 3	60.24
Kimi K2.5	42.51
Llama 3.3	27.74
DeepSeek V3.2	27.59
Grok 4.1	19.21
Gemini 2.5	16.83
GPT-OSS	14.42

F1 scores on SimpleQA Benchmark (Cranfield Evaluation)

Motivating study (Cranfield model)

- Fair share: market share you would expect — proportional to Cranfield benchmark capability (F1)
- Market Concentration: measure of skewness of share
  - Herfindahl–Hirschman Index (HHI): Square of ℓ_2 norm of shares
  - Effective number of Firms (EF): Reciprocal of HHI

Model	F1	Fair Share (%)	Fair Share (%) w/o Qwen3	Fair Share (%) w/o DeepSeek V3.2
Qwen 3	60.24	28.89	–	33.30
Kimi K2.5	42.51	20.38	28.66	23.49
Llama 3.3	27.74	13.30	18.71	15.33
DeepSeek V3.2	27.59	13.23	18.60	–
Grok 4.1	19.21	9.21	12.95	10.62
Gemini 2.5	16.83	8.07	11.35	9.30
GPT-OSS	14.42	6.91	9.72	7.97
HHI	–	0.18	0.19	0.22
EF	–	5.56	5.24	4.63

Lightly concentrated market

Heavily concentrated market

Motivating study (Marketplace Simulation)

Simulation Procedure

- Assume a fixed population of users, each with a preference distribution over systems
- Initialize uniform preference over systems
- Simulation steps (loop)
  - Sample k users
  - For each user
    - Sample a question
    - Select a system based on their pref. distribution
    - Observe the response from the selected system
    - Update pref. distribution based on the observed correctness

Model	Fair Share (%) w/o Qwen3	Fair Share (%) w/o DeepSeek V3.2
Qwen 3	–	33.30
Kimi K2.5	28.66	23.49
Llama 3.3	18.71	15.33
DeepSeek V3.2	18.60	–
Grok 4.1	12.95	10.62
Gemini 2.5	11.35	9.30
GPT-OSS	9.72	7.97
HHI	0.19	0.22
EF	5.24	4.63

Fair shares of **Lightly** and **Heavily** concentrated market based on **Cranfield**

Motivating study (Marketplace Simulation)

Simulation Procedure

- Assume a fixed population of users, each with a preference distribution over systems
- Initialize uniform preference over systems
- Simulation steps (loop)
  - Sample k users
  - For each user
    - Sample a question
    - Select a system based on their pref. distribution
    - Observe the response from the selected system
    - Update pref. distribution based on the observed correctness

1. *System ranking changes*
2. *Market shares are not proportional to benchmark scores*

Information that Cranfield CANNOT give you

Model	Fair Share (%) w/o Qwen3	Fair Share (%) w/o DeepSeek V3.2
Qwen 3	–	33.30
Kimi K2.5	28.66	23.49
Llama 3.3	18.71	15.33
DeepSeek V3.2	18.60	–
Grok 4.1	12.95	10.62
Gemini 2.5	11.35	9.30
GPT-OSS	9.72	7.97
HHI	0.19	0.22
EF	5.24	4.63

Model	Share	ΔFS
Kimi K2.5	29.40	(+0.74)
DeepSeek V3.2	21.00	(+2.4)
Llama 3.3	14.40	(-4.31)
Grok 4.1	13.80	(+0.85)
GPT-OSS	12.80	(+3.08)
Gemini 2.5	8.60	(-2.75)
HHI	0.19	0
EF	5.15	(-0.09)

Model	Share	ΔFS
Qwen3	51.60	(+22.71)
Kimi K2.5	14.80	(-5.58)
Llama 3.3	10.60	(-2.70)
Grok 4.1	9.60	(+0.39)
GPT-OSS	7.40	(+0.49)
Gemini 2.5	6.00	(-2.07)
HHI	0.32	(+0.10)
EF	3.15	(-1.49)

10 users, **100 simulation steps**,
5 users per step, 500 test queries in total
ΔFS: Fair share difference

Motivating study (Marketplace Simulation - Market Entry)

- We already pre-warmed the simulation with 6 models (t=1 - 100)
- Market Entry:
  From t=101, we introduce a 7th model, and let the simulation run (t=101 - 200) with the same set of questions.

- Two contrasting entry scenarios:
  - a strong model (Qwen3) enters a **lightly** concentrated market;
  - an average model (DeepSeek V3.2) enters a **highly** concentrated market.

Motivating study (Marketplace Simulation - Market Entry)

Qwen3 late entry.

Market Share (ΔFS) (%)	System Ranking	→	System Ranking	Market Share (%) (ΔFS)
Before Entry HHI=0.19 $t = (1 - 100)$			After Entry HHI=0.24 $t = (101 - 200)$	
		new	Qwen3	36.00 (+7.11)
(+0.74) 29.40	Kimi K2.5		Kimi K2.5	29.40 (+8.62)
(+2.4) 21.00	DeepSeek V3.2		DeepSeek V3.2	11.60 (-1.63)
(-4.31) 14.40	Llama 3.3		Gemini 2.5	6.60 (-1.47)
(+0.85) 13.80	Grok 4.1		GPT-OSS	6.40 (-0.51)
(+3.08) 12.80	GPT-OSS		Llama 3.3	5.60 (-7.7)
(-2.75) 8.60	Gemini 2.5		Grok 4.1	4.40 (-4.81)

DeepSeek V3.2 late entry.

Market Share (ΔFS) (%)	System Ranking	→	System Ranking	Market Share (%) (ΔFS)
Before Entry HHI=0.32 $t = (1 - 100)$			After Entry HHI=0.38 $t = (101 - 200)$	
(+22.71) 51.60	Qwen3		Qwen3	57.80 (+28.91)
(-5.58) 14.80	Kimi K2.5		Kimi K2.5	15.80 (-4.58)
		new	DeepSeek V3.2	12.0 (-1.23)
(-2.70) 10.60	Llama 3.3		Llama 3.3	4.60 (-8.70)
(+0.39) 9.60	Grok 4.1		GPT-OSS	4.20 (-2.71)
(+0.49) 7.40	GPT-OSS		Gemini 2.5	3.20 (-4.87)
(-2.07) 6.00	Gemini 2.5		Grok 4.1	2.40 (-6.81)

Realized shares ≠ expected benchmark "fair" shares.

- A strong model entering a lightly concentrated market
- **shock** to a market and **reshuffles** everyone
- An average model entering a highly concentrated market
- barely moves the market. Dominance continues, while **compressing** others.

Motivating study (Marketplace Simulation - Market Entry)

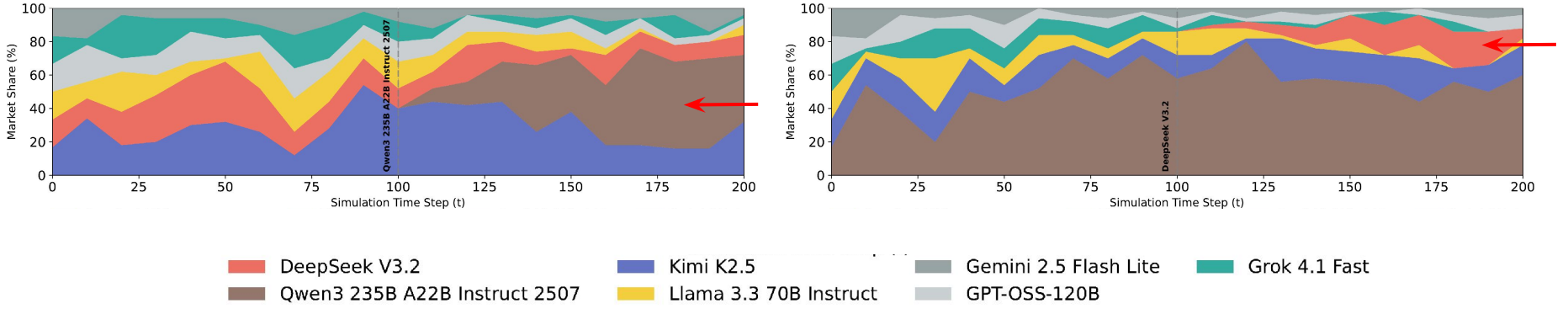
Qwen3 late entry.

Market Share (ΔFS) (%)	System Ranking	→	System Ranking	Market Share (%) (ΔFS)
Before Entry HHI=0.19 $t = (1 - 100)$			After Entry HHI=0.24 $t = (101 - 200)$	
		new	Qwen3	36.00 (+7.11)
(+0.74) 29.40	Kimi K2.5		Kimi K2.5	29.40 (+8.62)
(+2.4) 21.00	DeepSeek V3.2		DeepSeek V3.2	11.60 (-1.63)
(-4.31) 14.40	Llama 3.3		Gemini 2.5	6.60 (-1.47)
(+0.85) 13.80	Grok 4.1		GPT-OSS	6.40 (-0.51)
(+3.08) 12.80	GPT-OSS		Llama 3.3	5.60 (-7.7)
(-2.75) 8.60	Gemini 2.5		Grok 4.1	4.40 (-4.81)

DeepSeek V3.2 late entry.

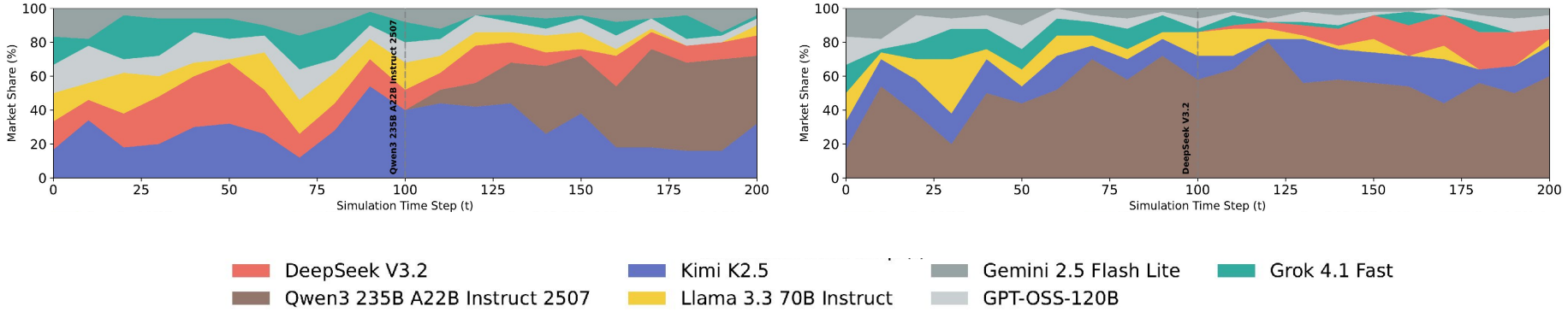
Market Share (ΔFS) (%)	System Ranking	→	System Ranking	Market Share (%) (ΔFS)
Before Entry HHI=0.32 $t = (1 - 100)$			After Entry HHI=0.38 $t = (101 - 200)$	
(+22.71) 51.60	Qwen3		Qwen3	57.80 (+28.91)
(-5.58) 14.80	Kimi K2.5		Kimi K2.5	15.80 (-4.58)
		new	DeepSeek V3.2	12.0 (-1.23)
(-2.70) 10.60	Llama 3.3		Llama 3.3	4.60 (-8.70)
(+0.39) 9.60	Grok 4.1		GPT-OSS	4.20 (-2.71)
(+0.49) 7.40	GPT-OSS		Gemini 2.5	3.20 (-4.87)
(-2.07) 6.00	Gemini 2.5		Grok 4.1	2.40 (-6.81)

Realized shares ≠ expected benchmark "fair" shares.



Motivating study (Lessons)

- Benchmark capability does not translate proportionally into market share
- Order of entry and market concentration changes who wins
- Dominance and winner-take-all effects emerge and this cannot be inferred from static benchmark alone
- A strong entrant can compress mid- and low-ranked models until the static ranking becomes uninformative
- **Need (simulated) evaluation that explicitly models interaction, adaptation, and competition.**



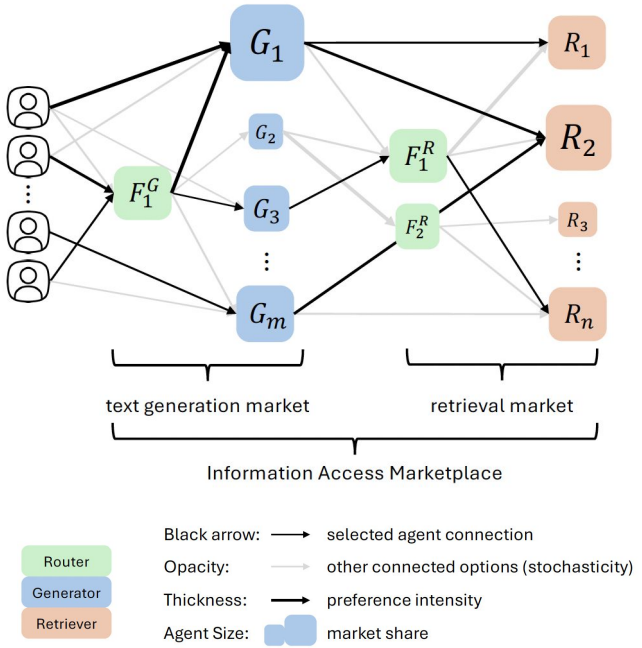
pip install marketplace-eval

- Graph-based simulation framework.
  - Node = market participants
  - Edge = preference-based selection(s)

- One example interaction:
  - user → router → generator → retriever
  - a judge scores the output as utility μ.

- Participants then update preferences from μ; repeated over many users and steps, traffic, preferences, and market share evolve. (parameters are all configurable)

- Focus on longitudinal outcomes:
  how agents attract demand, retain usage, and stay viable under competition.
  - Supports Market metrics
    - E.g., Customer retention, Dominance, Market Share, …



An example simulation snapshot of a RAG marketplace.
Users, generators, routers, and retrievers as participating stakeholders

pip install marketplace-eval

pip install marketplace-eval
https://github.com/kimdanny/marketplace-eval

```
marketplace_eval/       # Installable Python package (pip install marketplace-eval)
├── agents/             # Generator, router, retriever, and judge agent implementations
├── core/               # Node and link primitives plus IO helper nodes
├── humans/             # User taxonomy configurations and automatic profile generation
├── post_simulation/    # Market share, CRR metrics, plotting, and analysis CLI
├── prompts/            # Prompt templates for LLM judges and user simulation
├── system/             # System orchestrator, logging, types, and user population
└── utils/              # LLM client utilities for OpenAI-compatible APIs
```

- Simple configuration YAML
  - Configurations of simulation params, user population, update size frequency, marketplace graph, judge, etc.
- Highly flexible code extension.
  - You can override existing classes (update rules, routers, retrievers, and judges) without touching the core simulation engine.

What one simulation step does

- sample users → each picks a generator (over evolving preferences)
- generator calls routers / retrievers per graph topology
- Metric scores the answer → utility μ
- update preferences & router stats → log a StepRecord

Three Research Thrusts in the paper

- RQ 1: “How should a marketplace be simulated?”
  - Better user modeling (e.g., choice model and update model)
- RQ 2: “What metrics should be developed for marketplace evaluation?”
  - Agent-level metrics ; Market-level metrics
- RQ 3: “How to integrate marketplace eval. into evaluation campaigns?”
  - peer competition vs. benchmark competition
  - run submission vs. agent submission

A Long-Term Research Agenda

RQ1 · Simulation

- How should marketplaces be simulated?
- Model users: utility functions, choice models, evolving persistent preferences.
- Model agents: generators, routers, retrievers competing under quality–cost constraints.
- Couple dataset design with preference dynamics.

RQ2 · Metrics

- What metrics fit a marketplace?
- Agent-level: windowed market share, customer retention.
- Market-level: HHI concentration, dominance gap, expected-exposure / fairness.
- **Beyond Accuracy**

RQ3 · Adoption to Eval Campaigns

- How to fold into evaluation campaigns?
- Bring marketplace simulation to benchmarking.
- Axis 1: peer vs. benchmark competition.
- Axis 2: run submission vs. agent submission.
- Plus: validation & meta-evaluation of the simulation.

RQ1: Simulating the marketplace

How should a marketplace be simulated?
Model each stakeholder as a strategic actor — with a goal, an adaptation strategy, and market-level evaluation signals.

RQ1.1: Modeling users (demand side)

- **Objective:** maximize utility from interactions — answer quality, cost, and trust.
- **Utility function:** $U = \alpha \cdot \text{Quality} - \beta \cdot \text{Cost} - \gamma \cdot \text{Latency}$, with heterogeneous per-user sensitivities.
- **Choice model:** stochastic (softmax) selection over estimated utility.
- preferences are coupled to dataset design — topic-dependent vs. persistent, evolving users.
- behavioral effects (framing, exposure/network effects) can amplify market concentration.

RQ1.2: Modeling agents (supply side)

- **Generators:** attract & retain users under quality–cost tradeoffs; adapt retrieval depth and orchestration (no / single / multi / agentic retrieval).
- **Routers:** intermediaries allocating queries (user→generator, generator→retriever); exclusive vs. open routing.
- **Retrievers:** compete for repeated selection via domain specialization, personalization, and robustness; maximize marginal contribution.

Stakeholders as strategic actors

- Each stakeholder is defined by a functional role, not an architecture — the same LLM can be a generator or a retriever.
- Every stakeholder has a goal, an adaptation strategy, and market-level evaluation signals that complement accuracy.

Stakeholder	Goal	Adaptive Behavior in the Marketplace	Evaluation Focus
Users	Sustained utility from the marketplace over time	Adapt preferences over generators based on past satisfaction, cost, latency, or trust	Longitudinal utility, retention rate, switching behavior
Generators	Maximize user demand under quality–cost tradeoffs	Adapt interaction protocols and retrieval configurations to remain competitive	User utility, cost efficiency, market share, user retention
Routers	Efficient allocation of queries to downstream services	Exclusive routing (vertical integration) vs. open routing (market-wide discovery)	Counterfactual regret, allocation efficiency, diversity, and fairness
Retrievers	Remain selected by generators under competition	Compete via domain specialization, personalization, or robustness	Marginal utility contribution, selection frequency, long-term survival

primary stakeholders in an information access agent marketplace.

RQ2: Marketplace metrics

Accuracy stays foundational, but doesn't capture performance as a market participant. Two complementary classes of metrics, read off the simulation logs.

Agent-level metrics

Market Share (windowed) — fraction of traffic in a sliding window.

$$\mathrm{MS}_a(t;w) = \frac{\sum_{u\in U}\sum_{k=t-w+1}^{t} \mathbf{1}[q_u(k)=a]}{\sum_{u\in U}\sum_{k=t-w+1}^{t} 1}$$

Customer Retention — do users keep choosing an agent after first adoption?

$$\mathrm{CR}_a(m) = \frac{1}{|U_a|}\sum_{u\in U_a} \mathrm{CR}_{u,a}(m)$$

High market share does not imply high retention.

Marketplace-level metrics

HHI — market concentration; rises toward monopoly.

$$\mathrm{HHI}(t;w) = \sum_{a\in A} \mathrm{MS}_a(t;w)^2$$

Market Dominance — top agent's share above its fair share target ε*.

$$\Delta(t;w) = \max_{a\in A} \mathrm{MS}_a(t;w) - \epsilon_a^*$$

Expected Market Exposure — using market share as a proxy for exposure

$$\mathrm{EE}_{\mathrm{marketplace}}(t;w) = \|\mathrm{MS}_a(t;w) - \epsilon_a^*\|_2^2$$

RQ3: Adoption to evaluation campaigns

How can TREC / CLEF evolve beyond static, Cranfield-style run submission to capture marketplace dynamics?
Two orthogonal design axes.

Axis 1: Peer vs. benchmark competition

- **Peer:** submissions compete directly for traffic and exposure in one shared simulation
  - Can reveal systems' behavior by their dominance, concentration, and switching, etc.
  - Downside: outcomes depend on the year's participant mix.
- **Benchmark:** each submission competes against a fixed, organizer-provided simulated user/agent population
  - better reproducibility and clearer attribution.

Axis 2: Run vs. agent submission

- **Run submission:** participants provide static outputs that are treated as cached outputs; only the user population adapts
  - low cost and reproducible, but systems cannot adapt.
- **Agent submission:** participants provide executable agents that make sequential decisions under competition
  - enables adaptive routing and strategic behavior, but complicates stochasticity, reproducibility, and gives high computational burden to track organizers

Task design: target interaction-heavy tracks:

- **distributed IR (TREC Million LLMs)**
- **generator–retriever coordination (TREC RAG)**
- **interactive information access (TREC iKAT, CAsT).**