Toward Self-Evolving Recommenders: Agentic Feature Engineering in High-Throughput Systems

Anonymous Author(s)

Abstract

Feature engineering is a complex, labor-intensive bottleneck requiring extensive effort to extract predictive signals. To automate this, we present an industrial case study where LLM agents explore semantic feature spaces for click-through-rate (CTR) prediction in a massive real-time recommender. The system converts unstructured creative input into compact categorical features via an LLM enrichment library; pre-ranks thousands of candidates using an open-source tool; evaluates survivors via cold-start-aware AutoML; and deploys them through production training and serving pipelines. Deploying these semantic features across two production CTR models yielded relative offline RIG gains and online business lift. However, these gains incurred material serving costs, increasing inference latency by 30–50% and CPU time by 25–30%. We detail the architecture, evidence chain, and failure modes needed to transition agentic feature discovery into a deployable AI data scientist workflow.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Artificial intelligence**; *Multi-agent systems.*

Keywords

AI data scientist, LLM agents, feature engineering, recommender systems, CTR prediction, real-time bidding

1 Introduction

Online recommendation systems serve as an ideal, yet exceptionally demanding, testbed for automating data science workflows [2]. Because they are driven by direct, measurable user interactions, they provide the rapid feedback loops necessary to evaluate an *AI data scientist* system in the wild. However, shepherding a new idea into production requires surviving a rigorous process data extraction, feature materialization, model training, strict serving latency constraints, and live A/B testing.

In advertising recommenders, pipelines typically optimize click-through rate (CTR), the primary economic engine dictating ad ranking and revenue. While the space of potential CTR signals within unstructured creatives is vast, manual extraction is notoriously slow. Industrial CTR models heavily leverage high-cardinality feature crosses; yet, while architectures like DCN-v2 and AutoFIS efficiently learn these interactions [6, 16], they only optimize over available fields. Our agentic loop instead expands the field space itself by generating semantic feature specifications directly from unstructured creative data.

Recently, multi-agent LLM systems have proven highly effective at autonomously generating interpretable features from unstructured contexts [1, 7]. However, the strict real-time requirements of CTR prediction make querying foundation models in the live request path infeasible. To resolve this, our system shifts the heavy LLM inference offline using a closed-loop evolutionary paradigm: an LLM iteratively proposes feature specifications, an enrichment component materializes the values, and an evaluation stack scores them. While sharing conceptual similarities with early feature discovery methods [12], our core contribution is the production bridge: distributing this exploratory loop, productizing the pipelines, and validating the features in two live CTR models.

The paper makes three practical points. First, LLM-extracted features successfully surface predictive signals from unstructured creatives. By designing these as compact multi-value categorical fields rather than dense embeddings, they remain inspectable and easily fit existing feature-crossing machinery. Second, agentic search requires a two-speed evaluation stack: rapid pre-selection over thousands of candidates, followed by computationally heavy AutoML and A/B testing. Third, while online business wins were substantial even with modest offline lifts, they incurred material serving costs; thus, latency, CPU, and traffic effects must be rigorously evaluated alongside prediction metrics.

2 Related Work

LLMs for recommendation. Recent surveys show that LLMs can support recommenders through item/user representation, prompting, explanation, reranking, and industrial deployment patterns [15, 17]. Much of this work studies LLMs as recommenders or as representation engines. Our setting is different: the LLM is offline infrastructure for producing compact semantic fields, while the online recommender remains a conventional low-latency CTR model. Features, once discovered, are utilised in real-time inference via lookup based mechanism.

Automated feature engineering with LLMs. A growing line of research leverages LLMs for feature generation, predominantly on tabular datasets. Methods generate binary rules [4], apply Python transformations to existing columns [1], or fine-tune models via DPO [5]. Some also introduce iterative optimization using decision-tree reasoning [1, 7]. However, these methods rely on combining already-structured inputs. Conversely, when applied to unstructured text, LLMs typically extract opaque dense embeddings in a single, non-iterative pass [10]. Our work bridges these domains by iteratively optimizing over unstructured creatives to explicitly output transparent categorical fields rather than embeddings.

Agentic data science. LLM-agent work provides the tool-use and feedback-loop substrate [11, 14, 18]. Agent0 applies this idea to multi-value feature discovery for recommenders [12]. We build on that direction but focus on the missing production layer: batch enrichment, human approval gates, online A/B evidence, and serving-cost guardrails.

Table 1: Related work overview.

Line of work	Distinction of this paper
LLMs for recommenders [15, 17]	LLMs run offline to emit compact categorical fields for an existing low-latency CTR model, rather than serving as the ranker or retriever
CTR feature search [6, 16]	The agent expands the field space with semantic specifications over creative assets, rather than only selecting crosses over existing fields
General LLM agents [14, 18]	Tool use, memory, and reflection are constrained by feature contracts, production training, human gates, and online guardrails
Agent0-style discovery [12]	The system adds distributed execution, batch enrichment, OutRank preselection, A/B evidence, and serving-cost analysis

3 System architecture

Figure 1 summarizes the system.

Data Flow and Orchestration. The system ingests ad creatives and request contexts (e.g., text, URLs, image descriptors). Driven by an agentic **Architect-Executor framework**, an architect model (`GPT 5.4`) proposes novel features while an executor (`GPT 4.1 nano`) extracts them. The output augments the original data with structured categorical fields. Rather than using opaque embeddings, these fields decode visual and structural semantics—such as an ad's core value proposition—transforming unstructured marketing intuition into discrete, actionable and interpretable recommender signals.

The Two-Speed Agentic Research Loop. To efficiently discover and validate these signals, the exploration loop operates at two distinct evaluation speeds:

- **The Fast Path:** TThe architect generates thousands of candidate features. These are rapidly filtered in minutes using an mRMR-style approach [8, 9] via the OutRank library [13]. Candidates are ranked based on their mutual information (MI) with user clicks and redundancy against already-selected fields. To ensure fair comparison, MI scores are normalized against the feature's cardinality.
- **The Slow Path:** Survivors from the MI-based pre-selection enter a direct model-in-the-loop evaluation phase. Each selected feature is appended to the baseline model, which is then fine-tuned. The system ultimately scores and validates these candidate features based on the performance lift they provide to the newly fine-tuned model.

Scalable Batch Enrichment. To prevent feature enrichment from becoming a computational and financial bottleneck, we implemented a scheduled, config-driven Python library. This unified pipeline replaces fragile, ad hoc scripts with a production-ready system that shares a single code path across offline research, in-memory workflows, and automated Airflow based feature emission.

Three core design choices ensure this infrastructure remains both scalable and reliable:

- **Dynamic Schemas:** The library converts feature specifications into strict structured outputs. Depending on the feature, the executor uses single-value classifiers to extract simple value/confidence pairs, or multi-value extractors that yield multiple normalized categorical values, explicitly associated with individual confidence scores.
- **Multi-Axis Scaling:** To aggressively reduce overhead, throughput is scaled via deduplication of repeated data, batch prompting [3], and batch API calls. Together, these optimizations reduced batch processing costs by two orders of magnitude – from thousands of dollars per day to tens of dollars.
- **Operational Resilience:** To avoid the failure modes common to notebook-scale prototypes, the system is fortified with robust error-handling mechanisms, including token refreshes, fallback models, direct-call fallbacks for degraded batches, unmapped row retries, deadline enforcement, circuit breaking, and stale-file sweeps.

Human Gates and Online Promotion. Transitioning features to live traffic requires a rigorous, non-autonomous promotion process. Data scientists first audit prompts for leakage and review offline metrics. Upon approval, offline LLM outputs are flattened into standard multi-value categorical features, retaining only high-confidence labels. Finally, automated checks verify coverage, cardinality drift, and hashing compatibility, ensuring compliance with strict production latency and stability constraints.

4 Evaluation Framework

4.1 Experimental Setup and Metrics

We evaluate the agentic workflow across three distinct levels: offline model performance, online business impact, and system-level serving costs.

Offline Metrics. Offline, candidate features are evaluated using relative information gain (RIG). For binary clicks y_i and predictions $\hat{p}_i$, RIG is formally defined as:

$$\mathrm{RIG} = 1 - \frac{\mathrm{CE}(\hat{\mathbf{p}}, \mathbf{y})}{\mathrm{CE}(\bar{p}, \mathbf{y})},$$

where CE denotes the cross-entropy loss and $\bar{p}$ is the empirical CTR. Candidate features are also judged through feature ablations, calibration checks, feature-vector validation errors, and cold-start AUC (evaluated in the early hours after a new creative signature appears in the feature store).

Online Business KPIs. Online experiments compare control and variant serving stacks under the production budget-split A/B framework, using stable randomization units and pre-specified primary and guardrail KPIs. We monitor core system metrics, including a margin proxy (a per-mille revenue-minus-media-cost measure), gross revenue (GR), clicks, conversion-value utilization (CVU) relative to spend, and CPA behavior. We also track downstream traffic-type metrics and budget transfer effects.

System and Operational Metrics. Because the recommendation system operates under strict real-time constraints, we rigorously monitor system health. Key metrics include inference latency percentiles, CPU time per ad, timeouts, and client error rates.

Note on Attribution and Anonymization: We report all lift values as relative changes against control models, omitting or bucketing exact traffic and absolute metrics for anonymity. For deployments bundling LLM features with other updates, attribution is weaker; we therefore rely on offline ablations and segment slices to isolate specific effects.

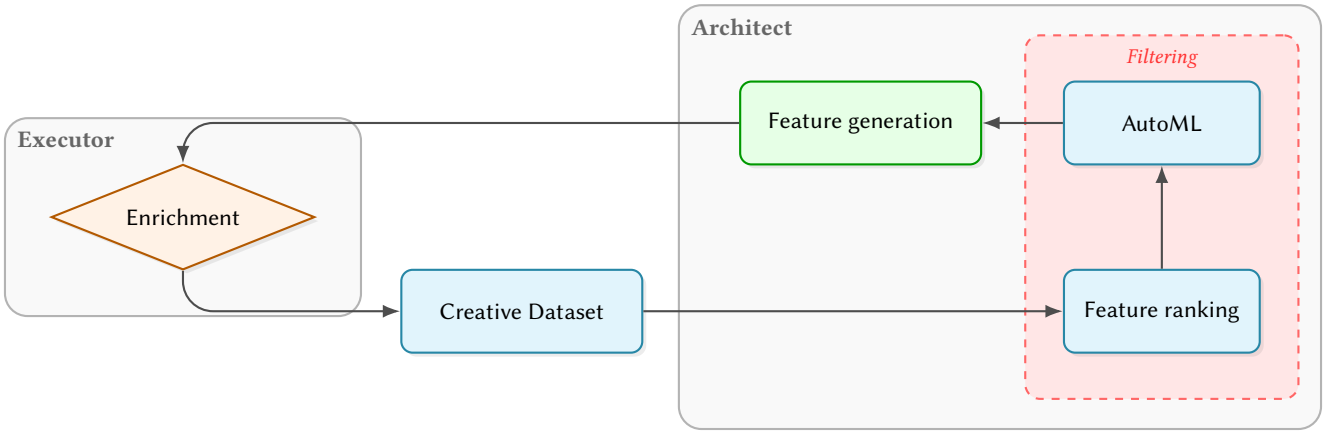

Figure 1: Agentic recommender workflow. An architect proposes features, an executor evaluates them, and feedback drives the search loop.

Table 2: Agentic search protocol and evidence funnel. Counts are bucketed.

Stage	State or artifact	Funnel	Gate
Candidate generation	JSON specs: inputs, task, examples, output schema, confidence rule, leakage checks	10^3 specs	Reject leakage, unsafe semantics, ambiguity, high cardinality
Batched enrichment	Typed config, dynamic schema, cache key, coverage/null/cardinality report	50+ enriched	Fail closed on invalid outputs; retry unmapped rows
Fast preselection	OutRank relevance/redundancy score within cardinality bucket	10^1–10^2 survivors	Ranking is a filter, not a deployment decision
Slow evaluation	Single-feature tests, from-scratch search, HPO, RIG/csRIG, validation errors	10^0–10^1 finalists	Require calibration, ablation, and validation-error checks
Production A/B	Versioned feature emission, serving variant, rollback trigger, online outcome	2 shipped	Monitor margin proxy, GR, CVU, CPA, latency, CPU, loss codes

Table 3: Two production deployments of LLM-derived semantic features.

	CTR (traffic type A)	CTR (traffic type B)
Experiment design	Production A/B; traffic bucket redacted	Production A/B; traffic bucket redacted
Promotion status	Passed internal KPI/guardrail gate; CI redacted	Passed internal KPI/guardrail gate; CI redacted
Main comparator	Production control plus semantic-field removal ablation	Production control; semantic fields checked against validation errors
Feature-count change	≈ +11%	≈ +8%
Representative semantic fields	6 fields	4 fields
Offline lift	positive relative RIG lift, promoted to online test	≈ +0.26% relative RIG
Cold-start evidence	AUC wins 12/13 early buckets; ≈ +0.1–0.3% relative	monitored; no validation regression
Online business lift	≈ +6% margin proxy; ≈ +5% CVU	≈ +6.6% GR; ≈ +6.3% margin proxy
Ablation evidence	Removing semantic fields reduced advantage; exact delta redacted	Semantic fields isolated from validation regressions; online ablation not disclosed
Calibration / CPA	similar; favorable CPA direction	similar calibration
Latency cost	≈ +50%	≈ +30–40%
CPU cost	≈ +30%	≈ +25%

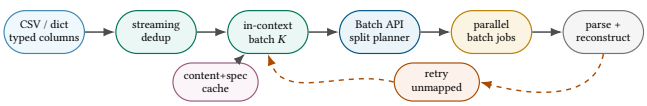

Figure 2: Productized batch enrichment. The pipeline minimizes API calls via deduplication, caching, and in-context batching, followed by parallel execution, schema validation, and full-cardinality reconstruction.

4.2 Production Deployments and A/B Test Results

CTR (traffic type A) This model used semantic features from creative text and description, and HPO at the online-learning-objective level. Offline search found a small set of features with a consistent positive relative RIG lift over production; the exact absolute RIG movement is omitted for anonymization. Online, the variant achieved about +6% on the margin proxy and +5% conversion-value utilization, with consistently higher clicks and revenue. Cold-start

analysis over the first hours of creative life was especially supportive: the variant won AUC in 12 of 13 hour buckets, with typical relative gains of roughly +0.1–0.3%. An ablation removing the LLM fields while keeping other changes reduced the advantage; the exact delta is redacted, but the direction was strong enough to keep the semantic fields in the promoted variant. Because this deployment also included request/context additions and HPO, we report it as strongest evidence for the end-to-end agentic workflow rather than as a perfectly isolated LLM-only treatment.

CTR (traffic type B) This added four production-ready LLM fields, most notably assumed result horizon – a field that attempts to estimate time horizon of the ad's "impact" (e.g., immediate, mid-term etc.). The offline movement was smaller, about +0.26% relative to the production model, but large enough to justify an online test. Online, traffic improved strongly: roughly +6.6% gross revenue and +6.3% on the margin proxy, with more clicks and revenue and similar calibration. Feature-vector validation error rates were similar to control. On the broader marketplace view, gains were not fully canceled by downstream engagement effects, yielding a positive aggregate marketplace result.

5 What We Learned

Offline evidence is necessary but not calibrated. MI preselection and offline evaluation were useful for pruning a huge feature space, but online effect sizes did not scale linearly with search scores. This was most visible for ad-side features: some offline lifts were modest, yet online business impact was large once the model interacted with bidding, budget, and cold-start traffic. Ultimately, this disconnect reveals a flaw in how autonomous AI systems are currently evaluated. Future benchmarks for AI data scientists should not merely score an agent's ability to maximize offline prediction metrics; they must also measure its ability to navigate the offline-to-online gap and select features that drive true production impact.

Compute is the main product trade-off. However, these predictive wins were driven by multi-value fields and complex feature crosses, which substantially increased serving complexity. For site traffic, inference latency rose by roughly 50% and CPU time by 30%, with timeout diagnostics indicating a +6.5% increase in critical loss over the control. Similarly, app traffic experienced a 30–40% increase in p95 latency across most regions, alongside a 25% rise in CPU time per ad. Because subsequent speed optimizations degraded the observed KPI lifts, the final production deployment accepted the higher latency overhead while strictly monitoring client errors. These material increases highlight the central tension of industrial-scale deployments. As emphasized by Anil et al. [2], modern recommendation systems are heavily bottlenecked by the "factory floor" realities of serving infrastructure. Consequently, offline RIG gains cannot be viewed in isolation; future iterations of the agentic loop must learn to optimize not just for predictive relevance, but explicitly for the strict computational budgets required to serve these features in real-time.

Agentic search benefits from distributed diversity, but can still converge prematurely. Distributing the architect-executor loop across multiple machines with shared memory significantly accelerated feature exploration and provided critical robustness against worker failures. However, this shared-memory architecture introduced a new challenge: when all parallel runs observed the same high-scoring prompts, the agents tended to prematurely converge on narrow regions of the semantic space. To counteract this mode collapse, we implemented explicit diversity constraints, cardinality-aware ranking, and a two-stage "fast-then-slow" evaluation protocol, which ultimately proved far more effective at discovering orthogonal signals than a single greedy optimization loop.

6 Limitations and Future Work

Limitations. Reproducibility is a primary limitation. Unlike public benchmarks, this study relies on proprietary datasets and KPIs, necessitating the redaction of exact A/B sample sizes, absolute metrics, and confidence intervals. Furthermore, responsible deployment introduces non-trivial risks (e.g., data leakage, sensitive inferences, harmful semantics). Consequently, the system is not entirely autonomous; it treats feature specifications as human-reviewable artifacts and strictly isolates LLMs from the online request path.

Future Work. These learnings suggest several future directions. First, meta-transfer HPO could share search experiences across CTR models, avoiding isolated searches. Second, the framework could expand beyond ad-side CTR to conversion rate (CVR) and impression-side features. Third, we aim to incorporate multimodal foundation models to process raw images directly, while carefully managing computational costs. Finally, our early, unsuccessful attempts at "semantic gradient" search—where agents use ranked features to iteratively guide new hypotheses—highlight an unsolved problem. While LLMs easily generate plausible initial hypotheses, mathematically optimizing over purely semantic feature spaces remains a formidable challenge.

7 Conclusion

We presented an industrial case study of agentic data science for recommender feature engineering. By combining LLM-generated semantic specifications, batched enrichment, OutRank preselection, AutoML, and production guardrails, the system selected features driving positive offline and online outcomes in two live CTR models. This experience suggests a practical definition of an AI data scientist: a tool-using loop proposing hypotheses, materializing data, learning from feedback, and remaining accountable to serving constraints.

References

[1] Nikhil Abhyankar, Parshin Shojaee, and Chandan K Reddy. 2025. Llm-fe: Automated feature engineering for tabular data with llms as evolutionary optimizers. *arXiv preprint arXiv:2503.14434* (2025).

[2] Rohan Anil, Sandra Gadanho, Da Huang, Nijith Jacob, Zhuoshu Li, Dong Lin, Todd Phillips, Cristina Pop, Kevin Regan, Gil I Shamir, et al. 2022. On the factory floor: ML engineering for industrial-scale ads recommendation models. In *Workshop on Online Recommender Systems and User Modeling*.

[3] Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model apis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*. 792–810.

[4] Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. 2024. Large language models can automatically engineer features for few-shot tabular learning. *arXiv preprint arXiv:2404.09491* (2024).

[5] Jeonghyun Ko, Gyeongyun Park, Donghoon Lee, and Kyunam Lee. 2025. Ferg-llm: Feature engineering by reason generation large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*. 4211–4228.

[6] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 2636–2645. doi:10.1145/3394486.3403314

[7] Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. 2024. Optimized feature generation for tabular data via llms with decision tree reasoning. *Advances in neural information processing systems* 37 (2024), 92352–92380.

[8] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence* 27, 8 (2005), 1226–1238.

[9] Erik Schaffernicht, Christoph Möller, Klaus Debes, and Horst-Michael Gross. 2009. Forward feature selection using Residual Mutual Information.. In *ESANN*, Vol. 1. 583–588.

[10] Zhiyi Shi, Junsik Kim, Davin Jeong, and Hanspeter Pfister. 2024. Surprisingly simple: Large language models are zero-shot feature extractors for tabular and text data. (2024).

[11] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Proceedings of NeurIPS*.

[12] Blaž Škrlj, Benoît Guilleminot, and Andraž Tori. 2025. Agent0: Leveraging LLM Agents to Discover Multi-value Features from Text for Enhanced Recommendations. arXiv:2507.18993 [cs.IR]

[13] Blaz Skrlj and Blaž Mramor. 2023. OutRank: Speeding up AutoML-based Model Search for Large Sparse Data sets with Cardinality-aware Feature Ranking. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1078–1083.

[14] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhe Wei, and Ji-Rong Wen. 2023. A Survey on Large Language Model based Autonomous Agents. arXiv:2308.11432 [cs.AI]

[15] Qi Wang, Jindong Li, Shiqi Wang, Qianli Xing, Runliang Niu, He Kong, Rui Li, Guodong Long, Yi Chang, and Chengqi Zhang. 2024. Towards Next-Generation LLM-based Recommender Systems: A Survey and Beyond. arXiv:2410.19744 [cs.IR]

[16] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of The Web Conference 2021*. ACM, New York, NY, USA, 1785–1797. doi:10.1145/3442381.3450078

[17] Lanling Xu, Junjie Zhang, Bingqian Li, Jinpeng Wang, Sheng Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2024. Tapping the Potential of Large Language Models as Recommender Systems: A Comprehensive Framework and Empirical Analysis. arXiv:2401.04997 [cs.IR]

[18] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of ICLR*.