Decoupled Agentic Pipelines: Optimizing Unit Economics for Automated Lead Generation

Rekson Bajimaya
Independent Researcher
Kathmandu, Nepal
Email: freelance@reksonbajimaya.com.np

***Abstract*—Traditional B2B lead generation often suffers from fragmented sources, rigid schemas, and the expense of hosted enrichment services. This paper presents a production-ready lead extraction pipeline optimized for local execution on an Apple M4 Pro system with 24 GB of memory. The architecture combines high-fidelity web scraping through Olostep, multi-stage filtering and deduplication in Google Sheets, and a cloud-backed ledger for persistent state management. On a batch of 350 candidate leads, composed of 200 custom software development leads and 150 mobile application development leads from New York and California, the pipeline reduced the raw set to 143 high-quality targets, extracted 90 valid contact email addresses using Gemini 3.1 Flash Lite Preview, and reduced the final output to 70 unique entries after spreadsheet-based deduplication. The system achieved a 96% request success rate and an average scraping latency of 12 seconds per successful lead. These results indicate that low-code orchestration and consumer-grade hardware can support efficient raw lead extraction at an approximate cost of $0.02 per qualified lead.**

I. Introduction

Automated B2B lead generation is frequently constrained by noisy web sources, fragile data schemas, and the high operating cost of hosted enrichment services. In practice, many prospecting workflows produce large volumes of low-value records that must be manually cleaned before they can support downstream qualification. These limitations are especially pronounced when the objective is to extract structured contact data from heterogeneous public web pages while maintaining traceability and low per-lead cost.

This study treats lead extraction as a systems-engineering problem: the question is not only how to collect contacts, but how much structured value can be recovered from noisy public web data when the pipeline must run locally and remain economically viable.

This paper examines a local, production-oriented extraction pipeline implemented on an Apple M4 Pro system with 24 GB of unified memory. The architecture combines Docker-based service isolation, n8n-based orchestration, Olostep-powered web scraping, Gemini 3.1 Flash Lite Preview extraction, and Google Sheets-based persistence, following the workflow described in [1]. Rather than treating the system as a generic automation script, the pipeline is organized as a modular data-processing workflow in which each stage performs a distinct function: raw acquisition, filtering and deduplication, semantic extraction, and final record consolidation.

The main contribution of this work is an empirically grounded account of how far a constrained local deployment can be pushed before quality or reliability degrades. On a batch of 350 candidate leads drawn from custom software development and mobile application development domains, the pipeline reduced the raw set to 143 filtered records, extracted 90 valid contact email addresses, and preserved 70 unique records after spreadsheet-based deduplication. These results provide a concrete benchmark for extraction quality, throughput, and unit economics under consumer-grade hardware constraints.

A second contribution is methodological. The paper documents the exact filtering, pacing, payload, and persistence controls used to stabilize the workflow, including regex-based source exclusion, duplicate suppression in Google Sheets, character-level payload limits, and rate-aware execution timing. These details are intended to make the pipeline reproducible and to clarify the tradeoffs between throughput, reliability, and final lead quality.

The remainder of the paper is structured as follows. Section II describes the methodology and system architecture. Section III reports the empirical results of the extraction run. Section IV concludes with a summary of the findings and directions for future extension.

II. Related Work

Transformer architectures established the attention mechanism as a scalable alternative to recurrent sequence modeling and became the basis for modern large language models [2]. Few-shot prompting then demonstrated that sufficiently large pretrained models can perform structured language tasks with minimal task-specific training [3]. Subsequent text-to-text and retrieval-augmented formulations extended this paradigm to information-intensive tasks, including summarization, extraction, and question answering [4], [5].

The present pipeline builds on these ideas but differs in its operational emphasis. Rather than optimizing for large-scale model training or benchmark performance, the system focuses on local execution, spreadsheet-backed persistence, and practical control of throughput and cost. This design choice is aligned with production extraction workloads in which the main challenge is not model pretraining, but the reliable transformation of noisy web content into structured lead records under strict hardware and unit-economics constraints.

III. Methodology

The pipeline is architected as a production-ready modular framework for automated B2B lead extraction on a local MacBook Pro M4 Pro environment with 24 GB of unified memory. The implementation combines Docker-based service isolation, n8n-driven event orchestration, and a Google Sheets-based cloud ledger for persistent state management and stakeholder visibility. The methodology is organized around a structured data processing pipeline in which raw web signals are transformed into filtered lead records suitable for manual review or later downstream activation. Each stage exposes deterministic outputs to the next stage, enabling reproducible execution under constrained local hardware resources.

A. System Architecture

The system utilizes a decoupled, event-driven architecture in which n8n workflows act as asynchronous triggers and coordination points. The execution stack is containerized with Docker, separating the scraping service, filtering logic, orchestration layer, and persistence interface into isolated runtime units. Raw prospect records are normalized into a common schema and persisted in Google Sheets, which functions as a cloud-integrated relational interface for lifecycle tracking and real-time review. This design supports low-friction inspection by non-technical stakeholders while preserving structured access for later automation. The high core count of the M4 Pro platform enables concurrent execution of scraping and filtering threads, reducing total processing time per lead and improving pipeline throughput under local deployment conditions.

B. Data Processing Pipeline

The data processing pipeline is implemented as a two-step modular workflow centered on raw acquisition and filter-based retention. The current study evaluates Stage I only: automated data ingestion, prospect discovery, cleansing, deduplication, and persistence of surviving records. Downstream enrichment and outreach are not part of the reported implementation and are therefore excluded from the empirical scope of this paper. Each operation uses a clearly defined intermediate schema, allowing the pipeline to remain extensible while preventing coupling between raw scraping logic and future engagement automation.

If the initial batch size is denoted by N_0, the filtered set by N_f, the extracted contact-email set by N_e, and the final unique set by N_u, then the observed pipeline can be summarized as

$$N_0 = 350, \quad N_f = 143, \quad N_e = 90, \quad N_u = 70. \tag{1}$$

The filtering retention rate is therefore

$$r_f = \frac{N_f}{N_0} = \frac{143}{350} \approx 0.409, \tag{2}$$

The extraction yield after Gemini 3.1 Flash Lite Preview processing is

$$r_e = \frac{N_e}{N_f} = \frac{90}{143} \approx 0.629. \tag{3}$$

The deduplication retention rate in Google Sheets is

$$r_u = \frac{N_u}{N_e} = \frac{70}{90} \approx 0.778. \tag{4}$$

The end-to-end yield from raw HTML input to final unique contact email is

$$r_{e2e} = \frac{N_u}{N_0} = \frac{70}{350} = 0.20. \tag{5}$$

Stage I integrates Olostep for high-fidelity web scraping and crawl acquisition. Candidate URLs are discovered through a crawler-scraper hybrid approach, after which targeted requests retrieve page-level content, metadata, embedded schema objects, and contact information. A multi-stage regex filtering layer removes low-value sources such as directory pages, SEO aggregation lists, and social media domains including Reddit and Facebook. Duplicate records are removed in Google Sheets using the formula $= \mathrm{SORTN}(\mathrm{Sheet1!A2{:}Z}, 9^9, 2, \mathrm{Sheet1!B2{:}B}, 1)$, which retains one row per email key and transfers the deduplicated output from Sheet 1 to Sheet 2. These controls reduce noise before persistence, improving the precision of the retained candidate set.

To improve collection quality, the ingestion layer applies lightweight filtering heuristics before persistence. Duplicate URLs, malformed entries, and irrelevant domains are suppressed early, and each record is tagged with source, timestamp, acquisition path, and crawl depth. This stage produces a prospect buffer that is sufficiently structured for downstream entity recognition while remaining faithful to the original source material.

The filtered output is capped at 30,000 characters per lead to preserve Google Sheets compatibility and to remain within practical context-window limits for downstream inspection. This payload limit also enforces compact record formatting, which reduces storage overhead and improves readability within the cloud ledger. To remain compliant with rate-limited free-tier API access, a randomized Wait node introduces controlled pacing in the range of 45–60 seconds per batch, thereby reducing the likelihood of 429 status codes during sustained execution. In addition, Continue on Fail behavior is enabled for 4xx and 5xx responses so that non-critical record-level failures do not halt the broader workflow.

For the extraction phase, the remaining HTML payloads are passed to Gemini 3.1 Flash Lite Preview, which performs constrained entity extraction, contact-email identification, and semantic summarization on the raw page content. The model output is reduced to a strictly typed JSON representation before being written to Google Sheets, enabling a direct mapping between parsed contact fields, description fields, and ledger columns. The first sheet stores the extracted email candidates and description records, while the second sheet stores the deduplicated final set.

C. Cost-Efficiency Analysis

The target unit cost of \$0.02 per qualified lead is achieved through a combination of token optimization, workflow mini-

mization, and infrastructure efficiency. The total cost per lead can be expressed as

$$C_{total} = \frac{\sum_{i=1}^{N_{leads}} (C_{scraping,i} + C_{tokens,i} + C_{api,i})}{N_{leads}} \approx \$0.02 \tag{6}$$

In this architecture, C_{tokens} is effectively minimized because the present study terminates after scraping, filtering, extraction, and persistence, thereby avoiding outreach calls. The term $C_{api,i}$ captures the Google Sheets and orchestration overhead associated with record insertion, duplicate checks, and workflow control operations for each lead i.

Infrastructure cost is reduced by running the stack locally on a MacBook Pro M4 Pro and by using open-source gateway tools and orchestration components instead of bespoke hosted middleware. Docker containers consolidate service dependencies, while workflow orchestration is handled via a low-code automation engine to minimize latency and reduce implementation overhead. Google Sheets provides durable, externally visible persistence without requiring additional managed services. Together, these design choices reduce both direct cloud expenditure and operational complexity, making the per-lead cost target feasible for extraction-only processing.

D. Methodological Summary

Overall, the proposed methodology follows a modular pipeline that separates acquisition, filtering, and persistence into distinct but interoperable stages. This separation improves maintainability, supports asynchronous execution, and enables fine-grained control over cost, quality, and response handling. The resulting architecture is optimized for scalable raw lead extraction while remaining consistent with low-cost operational constraints.

E. Human-in-the-Loop (HITL) Refinement

Initial results indicated that raw, high-volume extraction from broad geographic sectors such as New York and California produced many tier-1 targets but low response rates. To optimize conversion, the pipeline was refined into a human-in-the-loop (HITL) model in which a human-curated URL list serves as the primary ingestion seed, focusing agentic extraction on high-intent boutique agencies. This adjustment reduces the breadth of the initial crawl in exchange for a higher-signal lead set, making the workflow better suited to personalized outreach and downstream qualification.

IV. Results

This section reports the empirical performance of the extraction-only pipeline on the local MacBook Pro M4 Pro deployment. A representative batch of 350 candidate leads, composed of 200 custom software development leads and 150 mobile application development leads from New York and California, was processed through the scraping, filtering, and persistence workflow. Following the application of the multi-stage regex filter and duplicate suppression logic, 143 records were retained in the Google Sheets ledger as high-quality targets, corresponding to a retention rate of approximately 40.9 percent and a filtration rate of approximately 59.1 percent.

TABLE I
SUMMARY OF EXTRACTION OUTCOMES

Stage	Count	Rate	Notes
Raw leads	350	100%	Input batch
Filtered records	143	40.9%	Post-regex filtering
Extracted emails	90	62.9%	Gemini 3.1 Flash Lite Preview
Final unique records	70	77.8%	Google Sheets SORTN pass

The downstream Gemini 3.1 Flash Lite Preview extraction step was then applied to the 143 retained HTML payloads. From this filtered set, 90 records yielded valid contact email addresses, producing an extraction yield of approximately 62.9 percent over the retained set. A subsequent Google Sheets deduplication pass, implemented with the SORTN formula over the email key column as a post-extraction step, reduced the 90 extracted records to 70 final unique entries, corresponding to a deduplication retention rate of approximately 77.8 percent and an end-to-end unique-email yield of 20 percent relative to the original 350 inputs.

Operational reliability was evaluated using request-level outcome statistics collected during the run. The system achieved a 96 percent request success rate, while server-side failures accounted for approximately 2.6 percent of requests. The remaining transactions were associated with non-fatal retries and workflow-controlled continuation paths enabled through the n8n failure-handling configuration.

Latency measurements indicate an average scraping latency of 12 seconds per successful lead. This processing time reflects the combined cost of scraping, filtering, and downstream payload preparation under the local execution environment. In aggregate, these results suggest that the architecture sustains practical throughput on consumer-grade hardware while preserving lead quality through aggressive pre-persistence filtering and constrained LLM extraction.

From an economic perspective, the measured batch behavior is consistent with the unit-cost target described in the methodology. The observed filtration and deduplication stages reduce unnecessary downstream processing, while payload capping and controlled API usage limit interface overhead. Together, these results support the claim that the extraction pipeline can deliver high-quality unique qualified leads at approximately \$0.02 per lead under steady-state operation, with outreach remaining a future extension rather than part of the current evaluation.

V. Economic Analysis

This section quantifies the unit economics of the extraction pipeline rather than the downstream outreach stage. The exported n8n workflows separate the system into data collection, AI extraction, outreach, reply checking, and error handling, but the present analysis isolates the collection and extraction layers because outreach is outside the measured scope. Because the workflow is restricted to local execution, open-source orchestration, and spreadsheet-backed persistence, the dominant

variable costs are model inference, API access, local compute, and marginal energy consumption. The design goal is therefore not to maximize raw throughput, but to minimize the cost of producing one final unique contact record.

A. Cost Decomposition

Let the batch size be denoted by N_0, the filtered set by N_f, the extracted contact-email set by N_e, and the final unique set by N_u. The batch-level operating cost can be expressed as

$$C_{\text{batch}} = C_{\text{scrape}} + C_{\text{filter}} + C_{\text{llm}} + C_{\text{sheet}} + C_{\text{orch}} + C_{\text{energy}}. \quad (7)$$

The corresponding unit cost per unique lead is

$$C_{\text{lead}} = \frac{C_{\text{batch}}}{N_u}. \quad (8)$$

This formulation is preferable for the present study because only unique leads are considered actionable outputs. In the observed batch, $N_u/N_0 = 70/350 = 0.20$, so four-fifths of the raw input is eliminated before final persistence. The early rejection of low-quality records is the principal economic lever, since the expensive Gemini extraction step is applied only to the 143 filtered records rather than to the full 350-item batch.

B. Quota-Constrained Operation

The workflow is deliberately paced to remain within free-tier API budgets. A randomized Wait node introduces controlled batching in the range of 45–60 seconds, which lowers the probability of rate-limit violations and reduces the risk of fee-bearing overflow behavior. In practice, this shifts the system from bursty execution toward quota-stable execution, which is economically preferable when the objective is to preserve predictable marginal cost rather than maximize instantaneous throughput.

The use of Docker, n8n, and Google Sheets further compresses fixed infrastructure costs. Docker isolates dependencies without requiring a managed server environment, n8n provides workflow orchestration without custom middleware, and Google Sheets serves as a low-friction persistence layer without the overhead of a dedicated database service. In the exported workflows, the data collection flow performs pre-filtering and HTML capture before persistence, while the AI extraction flow maps raw page content into contact email, description, and personalization fields before spreadsheet updates. As a result, the remaining cost structure is largely variable and tied to model inference and any quota-bearing API operations.

C. Economic Interpretation

The system prioritizes economic efficiency over raw volume. Under this design, the relevant question is not how many URLs can be scraped, but how much structured value can be recovered per unit cost after filtering, extraction, and deduplication. The observed 20 percent end-to-end unique-email yield implies that the pipeline is intentionally conservative: most of the savings come from preventing unnecessary downstream processing, not from maximizing raw scrape count.

The reported approximate cost of \$0.02 per qualified lead should therefore be interpreted as a marginal, extraction-stage unit cost under the observed workflow settings. That estimate is most defensible when viewed as the cost of producing one final unique contact record from a noisy public-web batch under constrained local execution and free-tier-aware pacing.

VI. Conclusion

This paper presented a production-oriented lead extraction pipeline designed for local execution on an Apple M4 Pro workstation with 24 GB of unified memory. The implemented workflow combined high-fidelity web scraping through Olostep, multi-stage regex filtering, Gemini 3.1 Flash Lite Preview-based extraction, and Google Sheets-based persistence and deduplication. Across a batch of 350 candidate leads, the system retained 143 records after filtering, extracted 90 valid contact email addresses, and reduced the final output to 70 unique entries after spreadsheet-based deduplication.

The observed results indicate that consumer-grade hardware can support a practical extraction pipeline when expensive downstream operations are constrained through filtering, payload capping, and controlled pacing. The measured 12-second average scraping latency and approximately 20 percent end-to-end unique-email yield reflect the tradeoff between throughput and data quality in noisy B2B prospecting environments. Although the current study stops at extraction and record consolidation, the architecture remains extensible for future enrichment and outreach modules.

Future work will shift the pipeline toward a human-in-the-loop (HITL) operating model in which a researcher first selects high-intent domains, after which the agentic stack performs deeper semantic extraction and personalization for outreach. The current results suggest that while high-volume raw extraction can surface a substantial number of leads, conversion quality is inversely related to batch size, indicating that smaller, more selective batches may yield better downstream response rates. This pivot would preserve the reproducibility of the extraction workflow while improving the signal-to-noise ratio of the final outreach list.

References

[1] R. Bajimaya, "Decoupled agentic pipelines: n8n workflow for automated lead generation," Zenodo, May 2026. [Online]. Available: https://doi.org/10.5281/zenodo.20103300

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33*, 2020.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W.-t. Yih, T. Rocktaschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems 33*, 2020.