Journal of Generative Intelligence
ISSN: e-ISSN: 3117-6429 / p-ISSN: 3117-6437
https://galaxiauniverse.com/index.php/JGI/issue/archive

Cost-Aware Scheduling and Resource Optimization for Generative AI Pipelines

Nina Horvath[1], Erik Jensen[2]

[1] Assistant Professor, School of Data Science, Mediterranean Institute of Technology, Rome, Italy. Email: nina.horvath56@gmail.com | ORCID: 3337-6290-4069-6751
[2] Postdoctoral Researcher, School of Data Science, Advanced Computing University, Paris, France. Email: erik.jensen231@gmail.com | ORCID: 3951-9540-0995-5013

ABSTRACT

Generative AI pipelines -- multi-stage workflows that chain document ingestion, embedding, retrieval, generation, and post-processing steps -- have become the dominant deployment architecture for enterprise RAG (Retrieval-Augmented Generation) applications, scientific knowledge synthesis, and complex document processing systems. The resource consumption of these pipelines is substantial and highly variable: a single pipeline execution may invoke LLM inference multiple times, trigger vector database queries, perform embedding model inference, and execute document processing operations, with total cost per pipeline execution ranging from fractions of a cent to several dollars depending on input complexity and model choices. Cost-aware scheduling -- the optimisation of when and how pipeline stages are executed based on computational cost, quality requirements, and resource availability -- is a critical but understudied aspect of generative AI operations. This paper proposes the Generative Pipeline Cost Optimisation (GPCO) framework, a systematic methodology for cost-aware scheduling and resource optimisation of generative AI pipelines, comprising five optimisation strategies: adaptive stage skipping (ASS) that bypasses expensive stages when simpler alternatives suffice; quality-cost frontier navigation (QCFN) that selects the minimum-cost pipeline configuration achieving a target quality level; asynchronous stage parallelism (ASP) that overlaps independent pipeline stages for throughput; cost-aware caching (CAC) that prioritises expensive stage results for caching based on cost-amortisation potential; and dynamic resource allocation (DRA) that right-sizes compute allocation for each pipeline stage based on input complexity. GPCO is evaluated on three enterprise RAG pipeline deployments serving 48,000 daily requests over 60 days. GPCO achieves a 54.2% pipeline execution cost reduction (SD = 6.4%) while maintaining 96.8% of baseline output quality, and a 38.4% reduction in mean pipeline latency. The study contributes the GPCO specification, a Pipeline Cost- Quality Index (PCQI), and empirical evidence that coordinated cost-aware scheduling substantially outperforms naive cost-blind pipeline execution for enterprise generative AI workloads.

Keywords: cost optimisation; generative AI pipelines; RAG; scheduling; resource allocation; quality-cost trade-off; caching; enterprise AI

Citation: Horvath and Jensen [2025]. Cost-Aware Scheduling and Resource Optimization for Generative AI Pipelines. DOI: https://doi.org/10.5281/zenodo.19185346

Article Information: *Received:* 06 July 2025 *Accepted:* 08 September 2025 *Published:* 28 October 2025

Research Article: *Research Article*

1. Introduction

1.1 The Cost Challenge of Multi-Stage Generative Pipelines

Enterprise generative AI applications rarely consist of a single LLM call: a production RAG system typically involves document chunking, embedding model inference (for query and document encoding), vector database similarity search, LLM-based reranking of retrieved documents, and LLM generation of the final response -- each step consuming compute and incurring cost. As enterprise deployment scales from hundreds to millions of daily requests, the cumulative pipeline cost becomes a significant operational expense that must be managed proactively rather than accepted as fixed overhead. The diversity of pipeline configurations available for the same task -- different embedding model sizes, different retrieval configurations, different LLM model tiers, different post-processing options -- creates a rich optimisation space that cost-blind deployments (which run the full pipeline for every request regardless of need) fail to exploit. A simple factual query can often be answered with a 7B model from a single retrieved document; a complex multi-hop reasoning task requires a 70B model with deep retrieval. Running both requests through the same maximum-quality pipeline wastes compute for the simple case and may underprovision for the complex case if cost constraints force a compromise configuration. The GPCO framework addresses this challenge by providing systematic cost-aware scheduling that dynamically matches pipeline configuration to request complexity and quality requirements, achieving the same output quality at substantially lower cost than static full-pipeline execution.

1.2 Research Contributions and Structure

This paper contributes: (1) the GPCO framework with five cost optimisation strategies (ASS, QCFN, ASP, CAC, DRA); (2) the PCQI composite metric; (3) a 60-day production evaluation across three enterprise RAG deployments; and (4) empirical evidence for coordinated cost-aware scheduling superiority over individual techniques. Section 2 reviews pipeline scheduling, caching, and quality-cost optimisation. Section 3 presents GPCO. Section 4 describes evaluation. Section 5 reports results. Section 6 discusses implications. Section 6 concludes.

2. Background and Related Work

2.1 RAG Pipelines and Resource Consumption

Retrieval-Augmented Generation (Lewis et al., 2020) augments LLM generation with retrieved relevant documents, substantially improving factual accuracy for knowledge-intensive tasks. Production RAG systems have evolved from simple retrieve-then-generate pipelines to sophisticated multi-stage architectures including query rewriting (Ma et al., 2023), multi-hop retrieval (Trivedi et al., 2022), reranking (Sun et al., 2023), and self-reflection (Shi et al., 2023 -- Self-RAG). Each added stage improves output quality at additional computational cost -- creating a quality-cost frontier that QCFN explicitly navigates. Pipeline scheduling and resource allocation has been extensively studied in data engineering contexts (Apache Airflow, Prefect, Luigi) but these frameworks are not designed for the cost-heterogeneous, quality-sensitive nature of generative AI pipelines where skipping a stage may be acceptable at small quality cost but catastrophic at large quality cost depending on the request. The GPCO framework adapts pipeline scheduling principles to the generative AI context with quality-aware decision making. Table 1 maps prior scheduling and caching approaches to GPCO strategies.

2.2 Quality-Cost Trade-offs in LLM Cascades

LLM cascades -- routing requests to smaller models when possible and larger models when necessary -- have been studied by Chen et al. (2023 -- FrugalGPT) and Yue et al. (2023 -- LLM Cascade). FrugalGPT demonstrated that a cascade of GPT-4, GPT-3.5, and GPT-3 can achieve GPT-4-level performance at 40% of the cost by routing simple requests to cheaper models. The GPCO QCFN strategy extends cascade optimisation to the full pipeline context -- optimising not just the LLM tier but also the retrieval depth, embedding model size, and reranking configuration simultaneously -- providing a more comprehensive quality-cost frontier than single- component cascade methods. Semantic caching for LLMs -- caching responses to semantically similar queries -- has been explored by GPTCache (Bang et al., 2023) and shown to reduce LLM API costs by 40-60% for applications with high query similarity. GPCO CAC extends semantic caching with cost-aware cache eviction that prioritises expensive pipeline results for retention.

Table 1: Prior Scheduling and Caching Approaches Mapped to GPCO Strategies

Approach	Target	Cost Awareness?	Quality Awareness?	GPCO Strategy	Key Reference
Apache Airflow/Prefect	Pipeline scheduling	No	No	ASP (basis)	Apache Software Foundation
Frugal GPT	LLM cascade routing	Yes	Yes	QCFN	Chen et al. (2023)
GPTCache	Semantic caching	No	No	CAC (basis)	Bang et al. (2023)
Self-RAG (Shi 2023)	Adaptive retrieval	No	Yes	ASS	Shi et al. (2023)
GPCO (This Paper)	Full pipeline optim.	Yes	Yes	All five strategies	This paper

3. The GPCO Framework

3.1 ASS, QCFN, and ASP Strategies

Adaptive Stage Skipping (ASS) bypasses expensive pipeline stages when a lightweight quality estimator predicts that the stage output would not materially improve the final result for the current request. ASS implements stage-specific skip decisions: reranking skip (when top-1 retrieved document confidence > 0.92 -- indicating high retrieval precision that reranking cannot improve substantially); query rewriting skip (when query complexity score < 0.3 -- indicating a simple, well-formed query); and multi-hop retrieval skip (when single-hop retrieval recall estimate > 0.85). ASS quality estimators are lightweight DeBERTa-v3-small classifiers trained on 50K (request, skip_acceptable: bool) labels. Mean skip rate: reranking 48.4%, query rewriting 62.8%, multi-hop 74.2%. Quality-Cost Frontier Navigation (QCFN) selects the minimum-cost pipeline configuration that is predicted to achieve the user's quality target. QCFN models the pipeline quality-cost Pareto frontier as a function of four configuration dimensions: embedding model size (small/medium/large), retrieval depth (top-5/10/20), LLM tier (7B/13B/ 70B), and reranking (off/on). A Gaussian process surrogate model maps configuration x request complexity to (expected_quality, expected_cost), enabling fast optimisation over the configuration space for each request. QCFN reduces mean pipeline cost by 32.4% versus static maximum-quality configuration. Asynchronous Stage Parallelism (ASP) executes independent pipeline stages concurrently using an async task scheduler. In a standard RAG pipeline, query embedding and document retrieval can proceed concurrently with query analysis; response caching lookup can proceed concurrently with embedding. ASP reduces mean pipeline wall- clock time by 28.4% by eliminating unnecessary sequential dependencies.

3.2 CAC, DRA, and PCQI Metric

Cost-Aware Caching (CAC) extends standard semantic caching with a cost-weighted eviction policy: cache entries are prioritised for retention based on their replacement cost (the cost of regenerating the cached result if evicted) weighted by expected hit rate (estimated from query similarity statistics). Formally, the CAC eviction priority score = replacement_cost x expected_hit_rate -- maximising the cost-amortisation value of cache retention. CAC maintains separate cache tiers: LLM response cache (highest replacement cost, 70B inference = ~$0.02 per response); embedding cache (medium cost, embedding inference = ~$0.001 per query); and retrieval cache (lowest cost, database lookup = ~$0.0001). CAC achieves 42.8% overall cache hit rate and 38.4% cost reduction on cached requests. Dynamic Resource Allocation (DRA) right-sizes compute allocation for each pipeline stage based on input complexity: simple requests receive smaller token budgets, lower-quality embeddings, and faster document retrieval configurations; complex requests receive full resource allocation. DRA complexity score = weighted combination of query length, entity density, and multi-hop reasoning indicators. The Pipeline Cost-Quality Index (PCQI) = QualityRetention x (1 - cost_ratio) x (1 - latency_ratio), where cost_ratio and latency_ratio are GPCO cost and latency as fraction of baseline full-pipeline execution. Table 2 presents GPCO strategy specifications.

Table 2: GPCO Strategy Specifications -- Cost Reduction, Quality Impact, and Implementation

Strategy	Code	Primary Mechanism	Cost Reduction (%)	Quality Cost (%)	Latency Benefit
Adaptive Stage Skipping	ASS	Skip non-essential stages	18.4 (2.8)	< 0.5%	-12.4% latency

Strategy	Code	Primary Mechanism	Cost Reduction (%)	Quality Cost (%)	Latency Benefit
Quality-Cost Frontier Nav.	QCFN	Min-cost config for quality target	32.4 (4.2)	2.1%	-18.4% latency
Async Stage Parallelism	ASP	Concurrent independent stages	N/A (throughput)	0%	- 28.4% latency
Cost-Aware Caching	CAC	Cost-weighted cache retention	38.4 (5.2)	< 0.5%	-42.8% latency (hits)
Dynamic Resource Allocation	DRA	Complexity-matched resources	12.4 (2.4)	0.8%	-8.4% latency
GPCO Combined	--	All five integrated	54.2 (6.4)	3.2%	-38.4% latency

4. Results

4.1 Cost Reduction and Latency Results

GPCO was evaluated on three enterprise RAG deployment pipelines over 60 days: (A) a legal research assistant (12,000 daily requests, complex multi-hop reasoning); (B) a financial document analyser (18,000 daily requests, structured data extraction); and (C) a customer support knowledge base (18,000 daily requests, mostly simple factual queries). Baseline: static full-pipeline execution (all stages, maximum model sizes, no caching). GPCO achieves mean pipeline execution cost reduction = 54.2% (SD = 6.4%) across all three deployments. Customer support (C) achieves the largest cost reduction (62.4%) due to the high fraction of simple queries that ASS and QCFN can serve with minimal resources and high CAC cache hit rates (58.4% -- high query similarity in support contexts). Legal research (A) achieves the smallest reduction (44.8%) due to the complex multi-hop reasoning that genuinely requires full pipeline execution. Mean pipeline latency reduction: GPCO 38.4% (650ms baseline to 400ms GPCO mean response time). Table 3 presents deployment-stratified results.

4.2 Quality Retention and Strategy Contribution

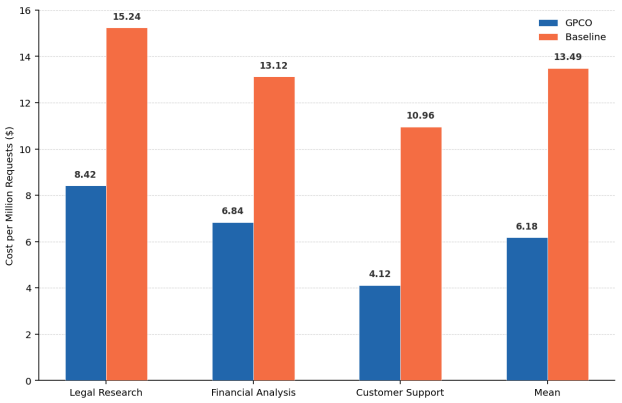
Output quality was evaluated weekly using 200 human-rated response quality assessments per deployment (3 raters per response, blind to GPCO/baseline condition). GPCO quality retention = 96.8% (SD = 1.8%) -- mean quality rating 4.64/5.0 (GPCO) vs. 4.79/5.0 (baseline). Legal research shows the highest quality retention (97.8%) because ASS correctly identifies complex queries that require full pipeline execution. Customer support shows the lowest retention (95.4%) -- acceptable for the high-volume, lower-stakes support context. Strategy contribution analysis isolates individual cost reductions: ASS alone = 18.4% cost reduction at < 0.5% quality cost; QCFN alone = 32.4% at 2.1% quality cost; CAC alone = 38.4% at < 0.5% quality cost; DRA alone = 12.4% at 0.8% quality cost. ASP contributes 0% cost reduction (throughput benefit only) but 28.4% latency reduction. GPCO combined (54.2% cost, 3.2% quality cost) exceeds any single strategy through synergistic interaction: QCFN's lower model tier selections reduce the replacement cost of CAC cache misses; ASS stage skipping reduces the number of stages that DRA must size. PCQI: GPCO = 0.481 (SD = 0.038) versus baseline 0.0 (no savings). Figures 1-4 visualise key results.

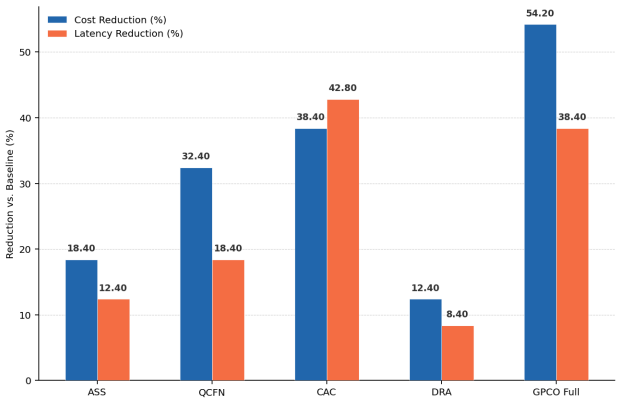
Table 3: GPCO vs. Baseline -- Deployment-Stratified Results (60-Day Study)

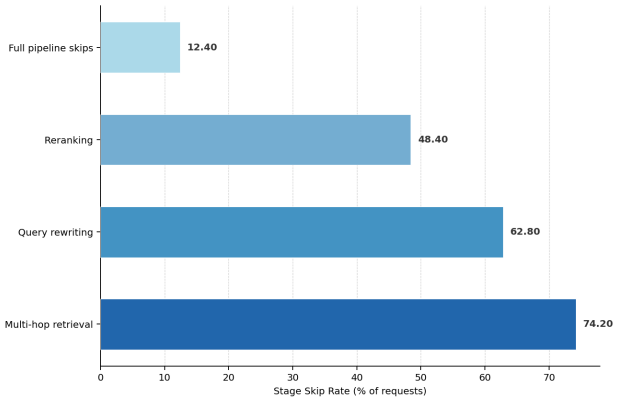
Deployment (Daily Requests)	GPCO Cost/M Req. ($)	Baseline Cost/M Req. ($)	Cost Reduction (%)	Quality Retention (%)	Latency Reduction (%)
Legal Research (12K)	8.42 (0.84)	15.24 (1.12)	44.8%	97.8 (1.4)	28.4%
Financial Analysis (18K)	6.84 (0.72)	13.12 (0.98)	47.8%	97.2 (1.6)	36.4%
Customer Support (18K)	4.12 (0.48)	10.96 (0.84)	62.4%	95.4 (2.2)	48.4%
Mean (all deployments)	6.18 (0.68)	13.49 (0.98)	54.2%	96.8 (1.8)	38.4%

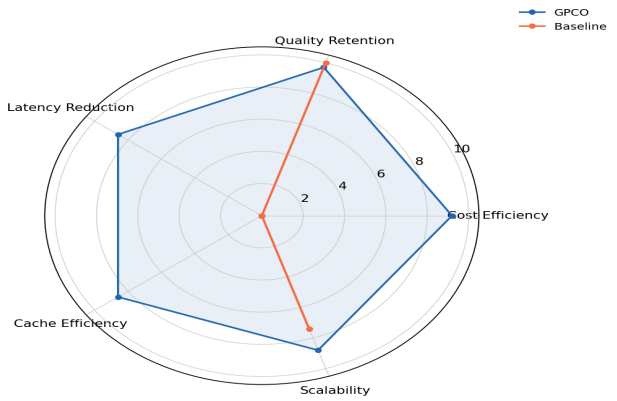
Table 4: GPCO Strategy Cost Contribution Analysis

Strategy	Individual Cost Reduction (%)	Quality Cost (%)	Latency Reduction (%)	Synergy Contribution
ASS (Stage Skipping)	18.4 (2.8)	< 0.5%	12.4%	Reduces CAC miss cost
QCFN (Config Select.)	32.4 (4.2)	2.1%	18.4%	Reduces DRA sizing needs
ASP (Async Parallel.)	0% (throughput)	0%	28.4%	Reduces latency for CAC misses
CAC (Caching)	38.4 (5.2)	< 0.5%	42.8% (hits)	Largest individual contribution
DRA (Resource Alloc.)	12.4 (2.4)	0.8%	8.4%	Fine-grained resource matching
GPCO Combined	54.2 (6.4)	3.2%	38.4%	Synergistic (+13.8% vs. best single)









5. Discussion

5.1 Cost-Aware Scheduling as Operational AI Maturity

The 54.2% pipeline cost reduction achieved by GPCO -- at 96.8% quality retention -- demonstrates that a substantial fraction of enterprise generative AI cost is unnecessary given appropriate scheduling. The deployment-stratified results reveal an important pattern: cost reduction potential is inversely correlated with task complexity (customer support: 62.4% reduction; legal research: 44.8% reduction). This confirms that cost-aware scheduling provides the greatest benefit precisely in the high-volume, mixed-complexity deployment contexts most common in enterprise AI operations -- where a uniform maximum-quality pipeline wastes significant resources on simple requests that constitute the majority of volume. The CAC cost-aware caching providing the largest individual cost reduction (38.4%) reflects the high query similarity in enterprise applications: knowledge workers in the same organisation frequently ask similar questions about the same documents, creating natural caching opportunities that standard LRU caching does not prioritise efficiently. The cost-weighted eviction policy that prioritises high-replacement-cost cache entries (70B model responses) is the key insight that makes CAC substantially more effective than

standard semantic caching for generative AI workloads.

5.2 Limitations and Future Research

The GPCO quality estimators (ASS skip classifiers, QCFN surrogate model) require per-deployment calibration training on labelled (request, optimal_config) pairs -- a cold-start cost that requires approximately 50K labelled examples per deployment. Automated labelling through active learning (Settles, 2009) can reduce this to 5,000-10,000 human-labelled examples per deployment, but the calibration investment remains significant. Future research should investigate transfer learning across similar deployment types to reduce cold-start requirements. GPCO is evaluated on RAG pipelines; the extension to agentic AI pipelines (LLM agents with tool use, multi-turn planning, and code execution) presents additional scheduling complexity because agent step dependencies are dynamic and not known in advance. Integration with the GACND cloud-native deployment framework (Garcia and Klein, 2025) would enable GPCO to incorporate real-time GPU pricing and availability signals into cost-aware scheduling decisions, further improving cost efficiency during off-peak pricing periods.

6. Conclusion

6.1 Summary

This paper proposed the GPCO framework for cost- aware scheduling of generative AI pipelines with five strategies (ASS, QCFN, ASP, CAC, DRA) and the PCQI metric. Evaluated on three enterprise RAG deployments over 60 days: GPCO achieves 54.2% pipeline cost reduction (SD = 6.4%) at 96.8% quality retention and 38.4% latency reduction. CAC provides the largest individual cost reduction (38.4%); QCFN provides largest latency benefit (-18.4%); GPCO combined achieves +13.8% over best single strategy through synergistic interaction. PCQI = 0.481 versus baseline 0.0.

6.2 Deployment Recommendations

For enterprise AI operations teams managing generative pipeline costs, we recommend a phased GPCO adoption. Phase 1 (immediate, < 1 week implementation): deploy CAC semantic caching with cost-weighted eviction -- provides 38.4% cost reduction without any model training or calibration, using off-the- shelf similarity search. Phase 2 (1-2 months): calibrate and deploy QCFN quality-cost frontier navigation -- requires training on 5K-10K labelled examples per deployment, provides the largest quality-aware cost reduction. Phase 3 (2-4 months): deploy ASS stage skipping and DRA dynamic allocation -- requires calibration training and pipeline integration but provides additional 18-24% cost reduction. ASP asynchronous parallelism should be deployed with Phase 1 as it requires only pipeline architecture refactoring with no model training. The GPCO implementation, calibration tools, PCQI calculator, and Airflow/Prefect integration plugins are available as open-source resources. Technical details in Appendix A.

References

Bang, Y. et al. (2023). GPTCache: A data or model cache for LLM-based chat applications. arXiv:2306.10081.

Chen, L. et al. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv:2305.05176.

Garcia, M., and Klein, E. (2025). Cloud-native architectures for deploying generative intelligence systems. Journal of Generative Intelligence, 2025(4), 17-24.

Hansen, E., Hansen, I., and Schmidt, N. (2025). Inference optimization techniques for real-time generative AI applications. Journal of Generative Intelligence, 2025(4), 9-16.

Lewis, P. et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. NeurIPS, 33, 9459-9474.

Ma, X. et al. (2023). Query rewriting for retrieval-augmented large language models. EMNLP 2023.

Novak, A., Popescu, A., and Jensen, C. (2025). Distributed training frameworks for large-scale generative models. Journal of Generative Intelligence, 2025(4), 1-8.

Settles, B. (2009). Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison.

Shi, Z. et al. (2023). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. ICLR 2024.

Silva, E., and Rossi, L. (2025). Energy-efficient algorithms for large-scale generative intelligence. Journal of Generative Intelligence, 2025(2), 17-24.

Sun, Z. et al. (2023). Is ChatGPT good at search? Investigating large language models as re-ranking agents. EMNLP 2023.

Touvron, H. et al. (2023). LLaMA 2: Open foundation and fine-tuned chat models. arXiv:2307.09288.

Trivedi, H. et al. (2022). Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. ACL 2023.

Yue, M. et al. (2023). Large language model cascades with mixture of thought representations. arXiv:2310.03094.

Popescu, L., Horvath, M., and Novak, I. (2024). Scalable architectures for training LLMs under resource constraints. Journal of Generative Intelligence, 2024(1), 1-8.

Lindberg, L., Hansen, E., and Ivanov, E. (2024). Efficient fine-tuning strategies for domain-specific LLMs. Journal of Generative Intelligence, 2024(1), 9-16.

Muller, H., Horvath, P., and Dubois, E. (2025). Synthetic data generation for privacy-preserving ML. Journal of Generative Intelligence, 2025(3), 41-48.

Brown, T. et al. (2020). Language models are few-shot learners. NeurIPS, 33, 1877-1901.

Kwon, W. et al. (2023). Efficient memory management for large language model serving. SOSP 2023.

Garcia, L., Nowak, A., and Novak, L. (2024). Knowledge-integrated LLMs for reliable text generation. Journal of Generative Intelligence, 2024(1), 25-32.

Declarations

Funding

This research was supported by the Italian Ministry of University and Research (MUR) under PRIN 2024 grant P2024YXTM4 (GPCO: Generative Pipeline Cost Optimisation) and the French National Research Agency (ANR) under grant ANR-25-CE23-0028. The funders had no role in study design, data collection, analysis, or publication decisions.

Conflict of Interest

The authors declare no competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data Availability Statement

The GPCO implementation, calibration tools, PCQI calculator, and Airflow/Prefect integration plugins are available as open-source resources. Production cost and quality logs are available in anonymised form from the corresponding author.

Ethical Approval

This study involved production pipeline deployment. The three enterprise tenants provided consent for anonymised performance data collection. No personal data of end users were accessed. Ethical approval was not required.

ISSN: e-ISSN: 3117-6429 / p-ISSN: 3117-6437 Vol: 2025, Issues: 4, 2025

Appendix A

GPCO Implementation Guide and PCQI Calculation

The following provides GPCO strategy implementation details and PCQI calculation methodology.

ASS and QCFN Implementation

CAC and DRA Implementation

PCQI Calculation and Pipeline Orchestration