LigaData

BARQ AI INSIGHTS · TECHNICAL WHITE PAPER

Conversational Analytics on Locally Hosted Language Models

Benchmarking 10 candidate models on a production-shaped analytics workload built entirely from synthetic and public data - and what it takes to run high-quality conversational analytics inside the customer's security perimeter.

86.7

quality index versus the production reference, from a fine-tuned 31B model

1,452

held-out output cells scored per candidate under anonymized judging

~6h

of GPU time measured to continue the adapter on an adjacent domain

Scope. Identical tool evidence is supplied to every model. The benchmark measures analytical generation and output-contract quality, not end-to-end tool selection. Selected-model tests used 10 simultaneous requests, without uniform coverage across the full model set.

Executive Summary

Enterprises in regulated industries increasingly want the analytical power of large language models without sending their data to third-party clouds. For many LigaData customers - including banks, telecom operators, and public-sector organizations - data residency is a hard requirement: request payloads may never leave their infrastructure.

Barq, LigaData's governed AI platform, is designed for this posture: governed AI on a semantic compute engine, deployed in the customer's tenancy. The remaining external dependency is the language model itself, and this paper reports how Barq can close it. We evaluated 10 candidate models - three distilled and fine-tuned by LigaData, two large unadapted open-weight models, and five frontier cloud APIs - plus the frontier model that serves as the production reference. The benchmark uses AI Insights' production prompts and six output contracts over synthetic and public data. Each candidate was scored on 1,452 held-out cells under an anonymized, rubric-referenced judging protocol.

The headline result: a 31-billion-parameter open-weight model, adapted with a compact 540 MB adapter, reaches a **quality index of 86.7 against the production reference while running on a single 80 GB GPU** at near-production latency. Its absolute rubric score is 6.64/10 versus 7.67/10 for the reference, and it outscores three of the five cloud APIs in this tested set. The strongest unadapted open-weight model reaches an index of 92.5, but requires a multi-GPU serving footprint. A follow-up study indicates that the same adapter transfers largely intact to a second, adjacent business domain and can be continued on that domain in roughly six GPU-hours.

For deployments where data cannot leave the customer's infrastructure, the results provide a practical range for the quality and hardware tradeoffs available today. This paper explains the platform context, what the benchmark does and does not measure, the results, the engineering decisions behind them, and the work underway to close the remaining gaps.

> **How to read the result.** This is a product benchmark, not a general-purpose LLM leaderboard. It measures analysis and generation under Barq's production output contracts while giving every model identical evidence. It does not measure end-to-end tool selection. Selected-model serving tests used 10 simultaneous requests, but coverage was not uniform across the model set, so this paper does not present concurrency as a cross-model benchmark. The 86.7 figure is a reference-relative quality index, not a percentage of questions answered correctly.

01 The Platform: Barq and AI Insights

1.1 Barq, briefly

Barq is LigaData's **governed AI platform**, built around one governed engine: **native semantic compute, governed data access and write, and a governed API** with a single policy authority in the execution path. It maintains live entity state and makes that state askable in plain language through chat and agents, or directly as queries. Structured and unstructured data live in one governed estate, allowing one cited answer across tables and documents. For the deployment pattern evaluated here, the full experience - including the LLM - is designed to run in the customer's tenancy.

At the heart of the platform is the **semantic compute engine**. Barq's semantic layer is not a descriptive catalog sitting beside the data - it is rich *and* executable: enriched metadata, business definitions, and policies are attached to semantics that the engine compiles and runs natively. Three layers translate business intent into executed analytics:

LAYER	WHAT IT DOES
Semantic Analytics Layer (SAL)	Interprets business intent. "Why did revenue drop?" resolves to a driver analysis with ranked contributions - an answer with interpretation, not a raw table.
Semantic Analytics Functions (SAF)	35+ composable analytics functions - aggregation, comparison, statistics, time series, clustering, simulation - carrying domain logic.
Semantic Functional Layer (SFL)	180+ foundational operations executing directly on the native compute engine, spanning build, analyze, monitor, act, and govern.

A single business question fans out into a composed call graph across these layers, compiled and executed natively with governance inline. This design has an important consequence for AI: the language model's job is to **select and parameterize validated, governed semantic functions**, not to author free-form SQL against raw tables. The semantic compute engine does the heavy computation under a single policy authority; the model is a focused, replaceable layer on top of it.

1.2 AI Insights: Barq's agentic analytics application

AI Insights is Barq's **agentic analytics app** - the application that puts the governed semantic stack behind natural language. It provides two families of capability:

- **NLP features** - users ask questions of their data in plain language and receive streamed analytical answers, charts, tables, and narrative interpretation, rendered in the Barq web UI or delivered through channels such as Slack and WhatsApp.
- **Agentic features** - the model plans and executes multi-step analyses: choosing which semantic functions to call, chaining tool calls over intermediate results, and composing the evidence into a final answer with recommendations and methodology explanations.

Every user question produces **six model outputs**: a streamed analytical answer plus five follow-up panels. Four of the six carry machine-readable or styled output contracts. A malformed chart specification or table payload is unusable by the renderer no matter how sound the underlying analysis, so format compliance is part of product quality and shapes the benchmark.

PANEL	OUTPUT CONTRACT	STRICT?
Main answer	Analytical prose, streamed	No
Chart	ECharts specification (JSON)	**Yes**
Table	Structured table hints (JSON/HTML)	**Yes**
Insights	Narrative interpretation (styled HTML)	Partial
Recommendations	Actionable steps (styled HTML)	Partial
Explanation	Methodology description (styled HTML)	Partial

Table 1. The six-panel AI Insights workload.

1.3 Why local models

Barq's deployment topology concentrates LLM processing in a single tier - the application server - with thin channel gateways upstream carrying no model logic. Swapping the model behind the application server leaves the rest of the system unchanged. The substitution surface is one tier wide, which makes local hosting a tractable engineering decision rather than a re-architecture.

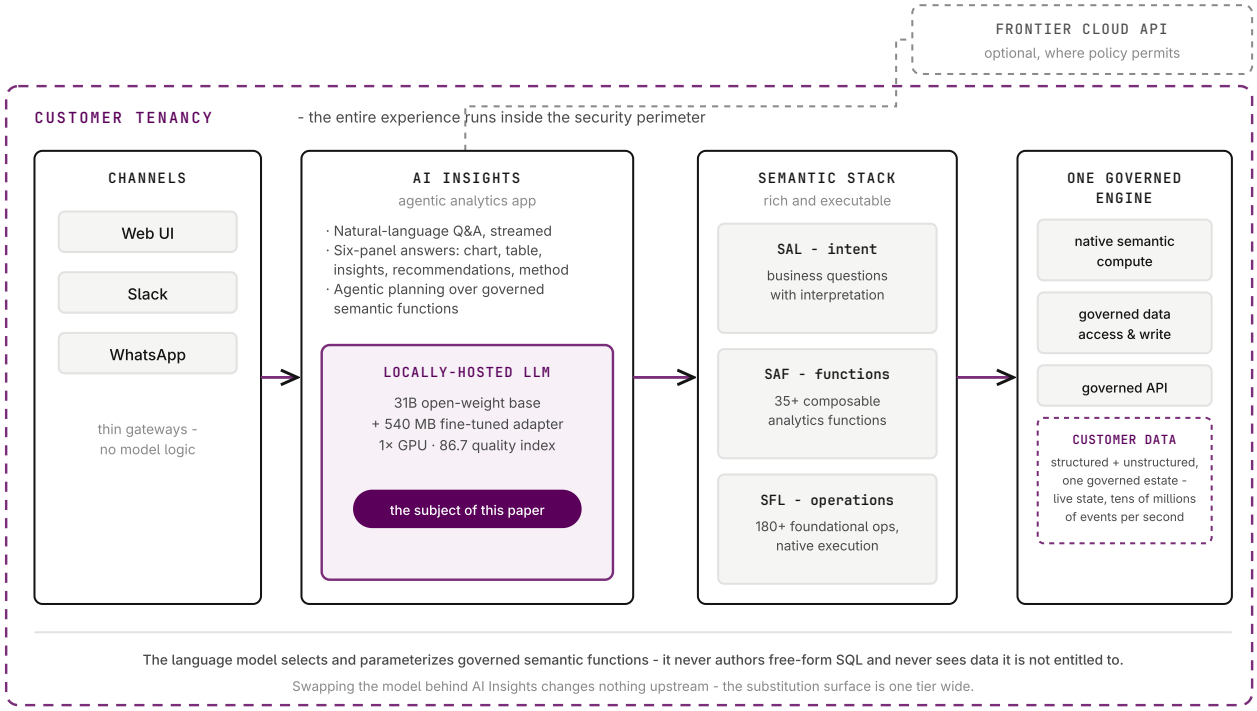

Figure 1. In the local deployment pattern, the channels, AI Insights app, semantic stack, governed engine, and model run inside the customer's tenancy. The optional frontier API path is available only where policy permits it.

The motivation is data residency, extending a principle already built into the platform: the governed execution path is designed to prevent the AI tier from seeing raw data outside its entitlement. Barq's metadata enrichment uses a firewall-segmented workflow: sensitivity is classified from schema, validated by a human, and confirmed-sensitive columns are masked before sample data moves. Sending each question and its supporting evidence to a hosted API could reintroduce an exposure that this design is intended to avoid. Local inference keeps the conversational tier inside the customer-controlled security perimeter.

There are economic motivations as well: per-token API spend scales with adoption, and hosted APIs expose customers to upstream pricing and deprecation decisions. Residency, however, is often the binding constraint. Where policy prohibits a hosted API, local inference is what makes this deployment pattern available.

02 Benchmark Design

2.1 Test set

The benchmark replays **242 questions** generated against a synthetic banking dataset and held out from all training. Each question runs through the production pipeline - production system prompts, output contracts, and rendering expectations - to produce all six panels: **1,452 scored cells per candidate**, with complete planned coverage.

Replay is **context-only**: every model receives byte-identical evidence from the reference run's tool results. This isolates analysis and generation quality from tool-selection variance and engine-state drift. The benchmark therefore evaluates the six customer-facing outputs, not the model's ability to choose and sequence semantic functions. End-to-end agentic tool selection is a separate workstream described in Section 7.

2.2 Judging protocol

Scoring uses a frontier judge model under a written rubric with three properties:

- **Anonymized** - the judge never knows which model produced a response.
- **Rubric-referenced and absolute** - each cell is scored against the supplied evidence and a cell-specific written ideal answer, not against a rival response.
- **Complete** - every planned cell is scored for every candidate; judging-harness faults are retried rather than charged to a model.

Each cell receives four scores on a 0–10 scale: **quality**, **correctness** (agreement with the supplied evidence), **completeness**, and **detail**, defined as useful specificity so verbose vagueness scores poorly.

Results are expressed as an index against the production frontier model: aggregate mean quality for the candidate divided by aggregate mean quality for the reference, multiplied by 100. The index makes product tradeoffs easy to compare, but it should not be read as a percentage of questions answered correctly. Close rankings should be treated as directional rather than as statistically distinct tiers.

2.3 Data provenance and benchmark scope

No client data was used in training or evaluation. The primary banking dataset is synthetic: it is production-shaped in schema and distribution but contains no real customer records. Synthetic questions were generated against that schema and replayed through the production pipeline, producing realistic workload traces without capturing customer traffic. The mortgage-applications study uses public-domain data unrelated to any LigaData client.

The benchmark is deliberately narrow. It measures output quality under AI Insights' production contracts, with tool evidence held constant. It does not claim to measure general reasoning ability, end-to-end agent behavior, or security certification. Separate serving tests exercised 10 simultaneous requests for selected models, but because coverage was not uniform, those runs are not used for cross-model concurrency claims. These boundaries make the result useful for the model-substitution decision addressed in this paper.

03 Benchmark Results

Model classes: FT = distilled and fine-tuned by LigaData, single GPU · OW = large open weights, unadapted, self-hostable · API = frontier cloud API · PROD = current production reference.

3.1 Overall quality

RANK	MODEL	CLASS	INDEX	QUALITY (0-10)
Ref	**Sonnet 4.6 (production)**	PROD	100.0	7.67
1	Sonnet 5	API	93.8	7.19
2	DeepSeek V4 Pro	OW	92.5	7.09
3	GPT-5.6 Luna	API	91.3	7.00
4	DeepSeek V4 Flash	OW	89.5	6.86
5	**Gemma 4 31B FT**	**FT**	**86.7**	**6.64**
6	GPT-5.6 Terra	API	84.0	6.44
7	Gemini 3.1 Pro	API	82.0	6.28
8	Gemini 3.5 Flash	API	79.8	6.12
9	Qwen3 MoE 30B FT	FT	68.1	5.22
10	Mistral 3.5 128B FT	FT	54.8	4.20

Table 2. Results for 10 candidates plus the production reference. The 86.7 index corresponds to an absolute quality score of 6.64/10 versus 7.67/10 for the reference. It is a relative benchmark index, not an accuracy rate.

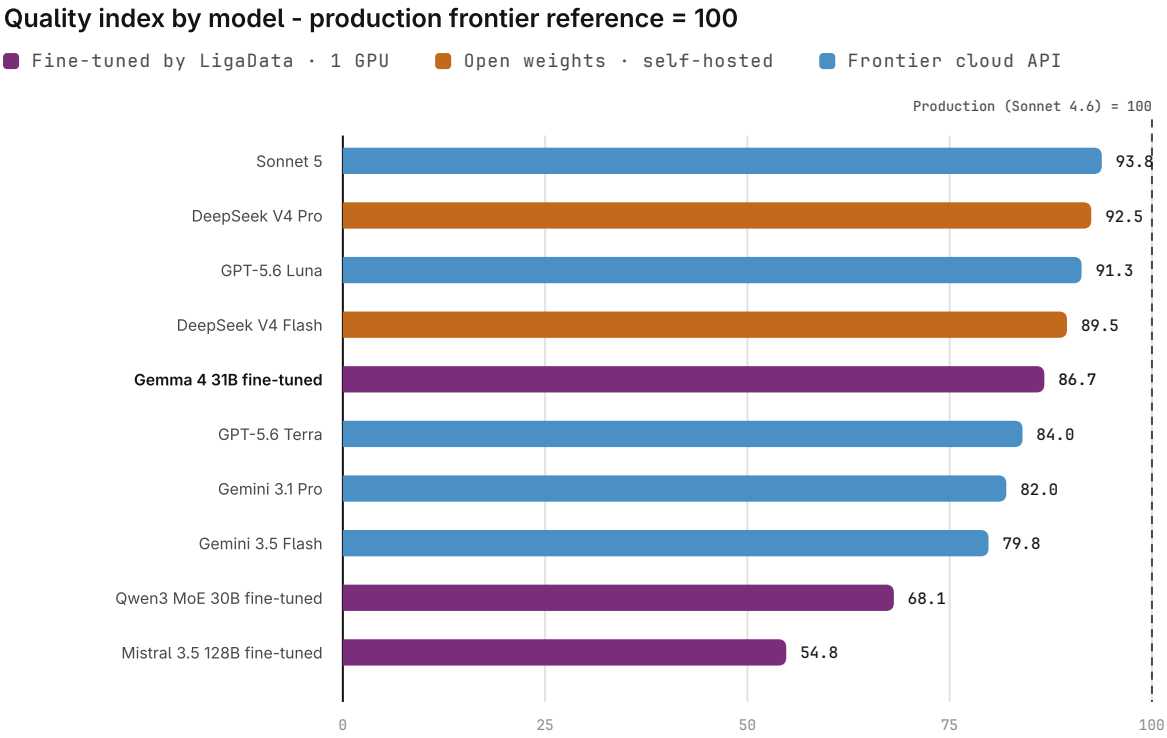

Figure 2. The fine-tuned 31B model sits above three of the five cloud APIs in this tested set and is the only candidate in the top five served on a single GPU. Rubric-referenced quality, 1,452 held-out cells per model, anonymized judging.

Three observations stand out. First, the distilled 31B model - running on **one 80 GB GPU** - outscores three of the five cloud APIs on this workload. Second, the highest self-hostable index is 92.5, from a large unadapted open-weight model with a multi-GPU footprint. Third, no candidate reaches parity with the production reference in this test.

3.2 Where the gap lives

Decomposing by panel shows the aggregate gap is not diffuse:

PANEL	PRODUCTION	SONNET 5	DEEPSEEK V4 PRO	GEMMA 4 31B FT
Main answer	8.39	7.26	7.78	7.53
Chart	6.91	6.75	6.52	5.81
Table	**7.28**	**5.73**	**4.76**	**5.04**
Insights	7.53	7.62	7.42	6.64
Recommendations	8.31	8.28	8.38	7.62
Explanation	7.57	7.52	7.69	7.22

Table 3. Absolute quality by panel (selected models). On insights, recommendations, and explanation, candidates reach or exceed the reference. The deficit concentrates almost entirely in the two strict-contract panels.

The panel results point to a focused improvement opportunity. Most of the observed gap sits in strict machine-readable artifacts, and chart generation is also the production reference's weakest panel. Notably, **the distilled 31B model beats both larger open-weight models on the table contract**, suggesting that exposure to the exact output format during training can partially substitute for scale. Section 7 describes the pipeline controls being developed around this boundary.

3.3 Latency and footprint

MODEL	CLASS	MEDIAN	× PRODUCTION	SERVING FOOTPRINT
GPT-5.6 Luna	API	3.4 s	0.14×	vendor cloud
Sonnet 5	API	16.4 s	0.68×	vendor cloud
Gemma 4 31B FT	**FT**	**28.3 s**	**1.17×**	**1× H100**
DeepSeek V4 Flash	OW	31.3 s	1.29×	284B MoE cluster
DeepSeek V4 Pro	OW	45.2 s	1.87×	460B MoE cluster

Table 4. Median single-request response time against production's 24.2 s (selected models). Separate exploratory tests used 10 simultaneous requests; model coverage was not consistently recorded, so no cross-model concurrency comparison is claimed. Among the self-hostable candidates, the distilled 31B model serves at approximately production latency on one card.

3.4 The two viable routes

The benchmark quantifies an exchange rate between the two self-hosting strategies:

	DISTILLED 31B (FINE-TUNED)	UNADAPTED 460B OPEN WEIGHTS
Quality index	86.7	**92.5**
Latency	**1.17×**	1.87×
Serving footprint	**1× GPU**	multi-GPU cluster
Table contract	**stronger**	weaker
Training cost	~$200 one-time	**none**
Adaptable to new formats	**yes**	no

Table 5. Neither route dominates. Where hardware footprint and format control matter most, distillation has the advantage. Where avoiding a training pipeline matters most, large unadapted open weights have the advantage. Barq supports both deployment patterns.

04 Cross-Domain Transfer: A Second Dataset

A benchmark on one workload leaves open whether the result is a property of the method or of that workload. We therefore ran a second study on an adjacent domain - mortgage applications - using public-domain data, a separate synthetic workload corpus, and a 1,506-cell held-out evaluation set.

The adapter transfers. With no mortgage-specific training, the banking adapter's benchmark index dropped by approximately five points on the adjacent domain under the same evaluation protocol. This is encouraging evidence of transfer, not yet evidence of broad cross-industry generalization.

Domain adaptation is fast and cheap. Continuing training from the existing adapter on 1,730 mortgage records took roughly six GPU-hours, compared with 15.8 hours for the original curriculum. It recovered approximately two-thirds of the observed domain-shift gap and improved all six panels in this study. The gains are consistent with a domain-knowledge gap rather than wholesale loss of the output formats.

The economics follow. Initial adaptation to a customer's workload is a one-time effort measured in hours on a single rented GPU. In this adjacent-domain study, continued adaptation cost a fraction of the first training run. A third internal experiment trained both domains together from the base model and informed the multi-domain consolidation approach described in Section 7.

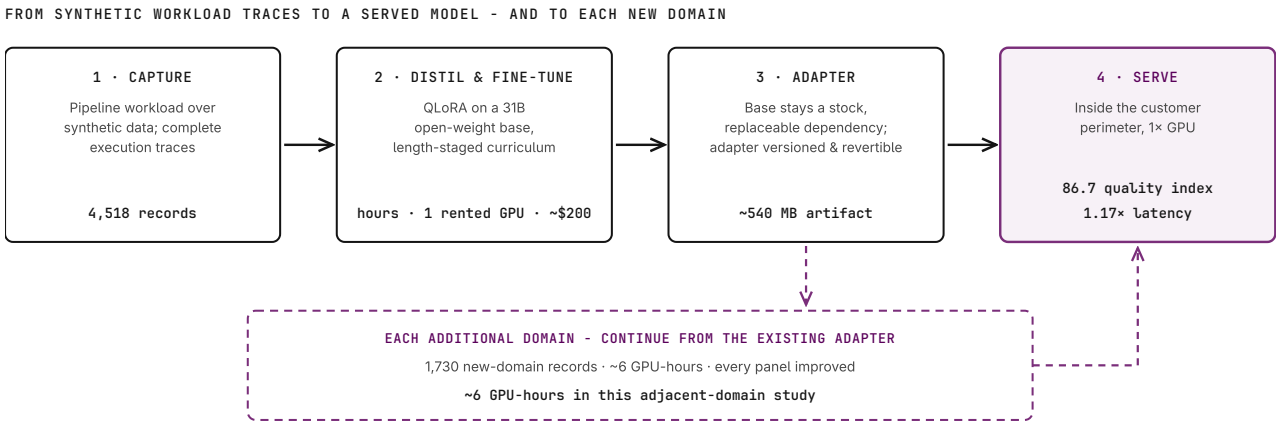

Figure 3. The adaptation pipeline. The first domain costs hours on one rented GPU; each additional domain continues from the existing adapter for a fraction of that.

Across both domains, evaluated with different frontier judges, the panel ordering is consistent: strict-contract panels are hardest, and the table contract is hardest of all. This supports the working hypothesis that a material part of the remaining gap is structural and can be addressed at the pipeline boundary. A planned multi-judge evaluation will test that conclusion more directly.

05 Engineering Findings

The numbers above reflect a set of engineering decisions that produced material improvements in internal trials. These are the most transferable lessons for repeating the process across customers and domains.

Distil from production-shaped synthetic traces, not hand-authored examples. Synthetic questions generated against the target schema are replayed through the production agent so that each training record contains a complete execution trace: tool calls, returned results, and the composed answer. This preserves the application's prompts, contracts, and awkward cases without using client traffic or customer records. The same capture process can then be pointed at a new synthetic or approved public dataset for another domain.

Supervise the answer, not the prompt. In this workload a prompt may run 15,000 tokens against a 500-token answer. Focusing the training objective on the answer span, rather than on text the model will never generate, produced the largest improvement observed in our internal training trials. The wrong objective can still show smooth loss curves, so the pipeline now validates the training labels and generated samples before an expensive run proceeds.

Base-model fit beats parameter count. Three bases were adapted under the same recipe, with corpus, curriculum, and hyperparameters held constant. The 31B dense instruction-tuned base reached an index of 86.7; a 30B mixture-of-experts base reached 68.1; and a 128B dense base reached 54.8. The operational lesson is to screen candidate bases cheaply before committing a full training run. Suitability for the workload can matter more than headline parameter count.

Compact adapters keep the system operable. Adaptation uses QLoRA. The base model stays frozen and replaceable, while workload-specific learning lives in a roughly 540 MB adapter that can be independently versioned, shipped, and reverted. In the measured setup, the quantized 31B base, adapter, optimizer state, and 16k-token activations fit within one 80 GB GPU, avoiding multi-GPU training complexity.

Audit distributions, not only statuses. The costliest operational failures were silent: a run could report healthy while producing degraded or empty output. Auditing response-length distributions caught failures that status checks missed. The pipeline now places low-cost assertions before expensive evaluation and training phases and monitors output distributions throughout each run.

06 Deployment Economics

Adaptation cost was low in this programme. Four full training runs (approximately 61 GPU-hours), self-hosted evaluation serving, and frontier API usage for corpus construction and judging cost less than $1,000 in the measured internal programme. The dominant expense was API usage for corpus construction and judging, not GPU time. These figures are historical internal estimates rather than supplier quotes.

Serving is a fixed-capacity cost. At the prices used in this study, one pilot-grade GPU pod cost about $77/day, or approximately $28,000/year if continuously provisioned. API pricing scales per token or query instead. Actual break-even volume depends on prompt size, answer length, utilization, concurrency, commercial terms, and operational support.

- Below the break-even volume, an API may be cheaper, and self-hosting may still be justified by residency.
- Above it, fixed-capacity serving can also have a cost advantage.
- Where policy prohibits a hosted API, local inference makes the deployment available regardless of the price comparison.

Plan for engineering time. Self-hosted models took two to four times longer to evaluate than API-hosted ones once pod provisioning, serving configuration, and operational debugging were included. Any self-hosting business case should account for that work; Barq's packaged serving stack is designed to absorb much of it for customers.

07 What Comes Next

The current results answer the model-substitution question for one production-shaped workload. The next workstreams extend coverage and remove the most important operational gaps.

One adapter, many domains. The next study consolidates multiple domains into one adapter and compares pooled training, rehearsal, sibling-adapter merging, and per-domain routing. The goal is to keep the cost of adding each new domain predictable.

Agentic tool selection with local models. A purpose-built benchmark will test whether a local model can select Barq semantic functions, parameterize them against a validated schema, and chain calls over intermediate results. This complements the generation benchmark reported here.

Contract-enforced structured output. Schema validation, automated retry, and constrained decoding are being added at the output boundary. These controls target the chart and table failure modes observed across model families without requiring more training compute.

Distant-domain transfer. Banking and mortgage are adjacent. Planned evaluations in telecom, infrastructure operations, and public-sector workloads will test how far the adaptation method travels.

Arabic and multilingual analytics. A bilingual Arabic-English corpus will extend the six-panel protocol across scripts, documents, and numeric conventions relevant to MENA deployments.

Evaluation across judge families. The next evaluation cycle will use multiple judge families, calibration checks, and explicit format-compliance rates alongside rubric scores.

Production serving at volume. Selected-model tests have exercised 10 simultaneous requests. The next standardized run will apply the same concurrency protocol across deployment candidates and report throughput, batching behavior, and tail latency, producing a sizing guide that maps tenant query volume to GPU capacity.

08 Conclusion

A 31B open-weight base, adapted with a 540 MB adapter distilled from production-shaped synthetic traces, reaches a quality index of 86.7 against the production reference while serving on one 80 GB GPU at 1.17 times the reference latency. Its absolute quality score is 6.64/10 versus 7.67/10 for the reference, and it outscores three of five cloud APIs in this tested set. The strongest unadapted open-weight model reaches an index of 92.5 with a larger serving footprint. In a second, adjacent domain, roughly six GPU-hours of continued training recovered most of the observed transfer gap.

Where a frontier API is permitted, local hosting is a quality, cost, and operational tradeoff whose magnitude can now be measured on the customer's workload. Where policy prohibits a hosted API, the results show that high-quality analytical generation can run on a predictable single-GPU footprint inside the customer-controlled environment.

The evidence indicates that a material share of the remaining gap sits in strict machine-readable output contracts. Contract validation, constrained generation, multi-domain consolidation, agentic evaluation, and bilingual benchmarking are the next steps in turning that finding into a broader production capability.

Evaluate this on your own workload. To discuss an on-premises AI Insights evaluation on your own workload, contact LigaData.

References

01 Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network.* arXiv:1503.02531

02 Kim, Y., & Rush, A. M. (2016). *Sequence-Level Knowledge Distillation.* EMNLP

03 Hu, E. J., et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models.* arXiv:2106.09685

04 Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). *QLoRA: Efficient Finetuning of Quantized LLMs.* arXiv:2305.14314

05 Wang, Y., et al. (2022). *Self-Instruct: Aligning Language Models with Self-Generated Instructions.* arXiv:2212.10560

06 Zheng, L., et al. (2023). *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena.* arXiv:2306.05685

LigaData