TrapCheck

Saving the world from bad Carbonara

It started with a friend's story from Rome.

He found a restaurant with a 4.7-star rating on Google Maps, right in a prime location.

The result? Expensive, terrible Carbonara.

A deeper look at the reviews revealed accusations of "fake reviews" and pressuring tourists for 5-star ratings.

This is a real problem: a high rating is no longer a reliable signal.

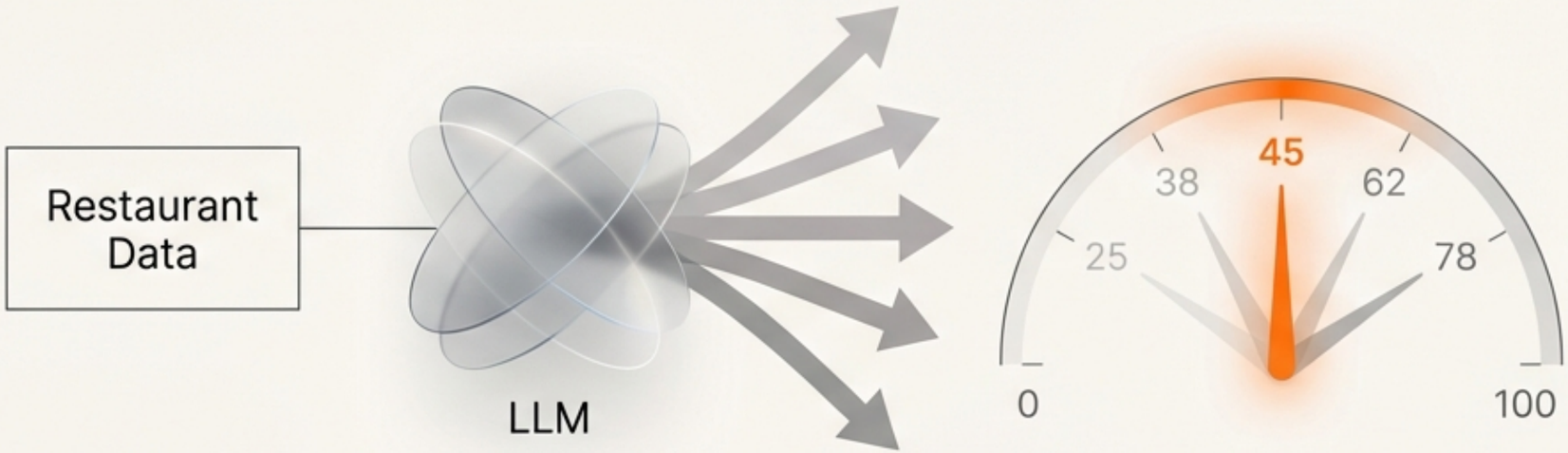
My first attempt was an inconsistent mess.

The MVP used Gemini 2.5 Flash to analyze Google reviews directly.

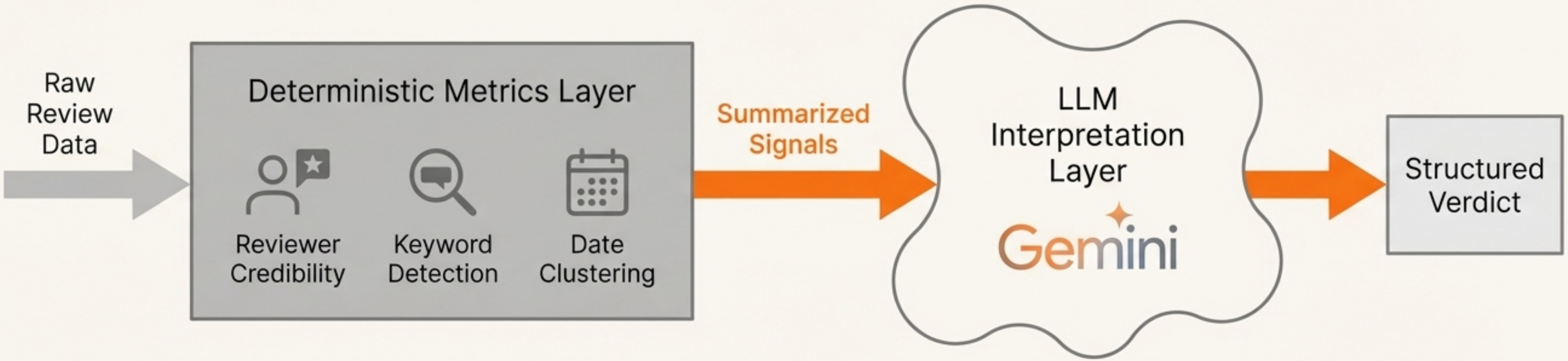
The problem: On the exact same data, results could vary by over 20 points.

This variance could flip the verdict from a “local gem” to a “possible trap.”

The core issue: We were asking an interpretive model to do a deterministic job.



The solution was to **separate signal from** interpretation.

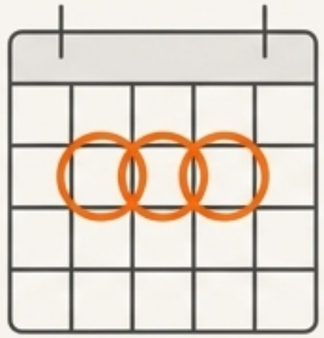


By computing quantitative signals *before* the LLM sees them, we achieve 100% signal stability on every run.

We built a system to detect specific manipulation patterns.

Credibility Inversion

A high-severity signal that fires when negative reviewers are demonstrably more credible (more reviews, photos, Local Guide status) than positive reviewers.



Review Clustering

Detects suspicious spikes of positive reviews on the same few days, indicating potential review manipulation campaigns.

Language Credibility Split

Flags when tourists (writing in English, etc.) dominate positive reviews, while locals (writing in the native language) dominate the negative ones.

But the system still struggled with the ambiguous “mixed” cases.

The metrics-first approach was perfect for clear-cut cases.

The new battleground was the “mixed” venues—places with conflicting signals, like a famous spot that is genuine but also has high prices.

The model needed more than just signals; it needed context.

To solve ambiguity, we gave the model a library of case files.

149

curated global venues.

50 / 50 / 49

Balanced Distribution: Traps, Gems, & Mixed cases.

Reddit, TripAdvisor, travel forums, and food blogs.

Diverse Sources.

Restaurants, museums, attractions, tours, and shops.

Venue Agnostic.

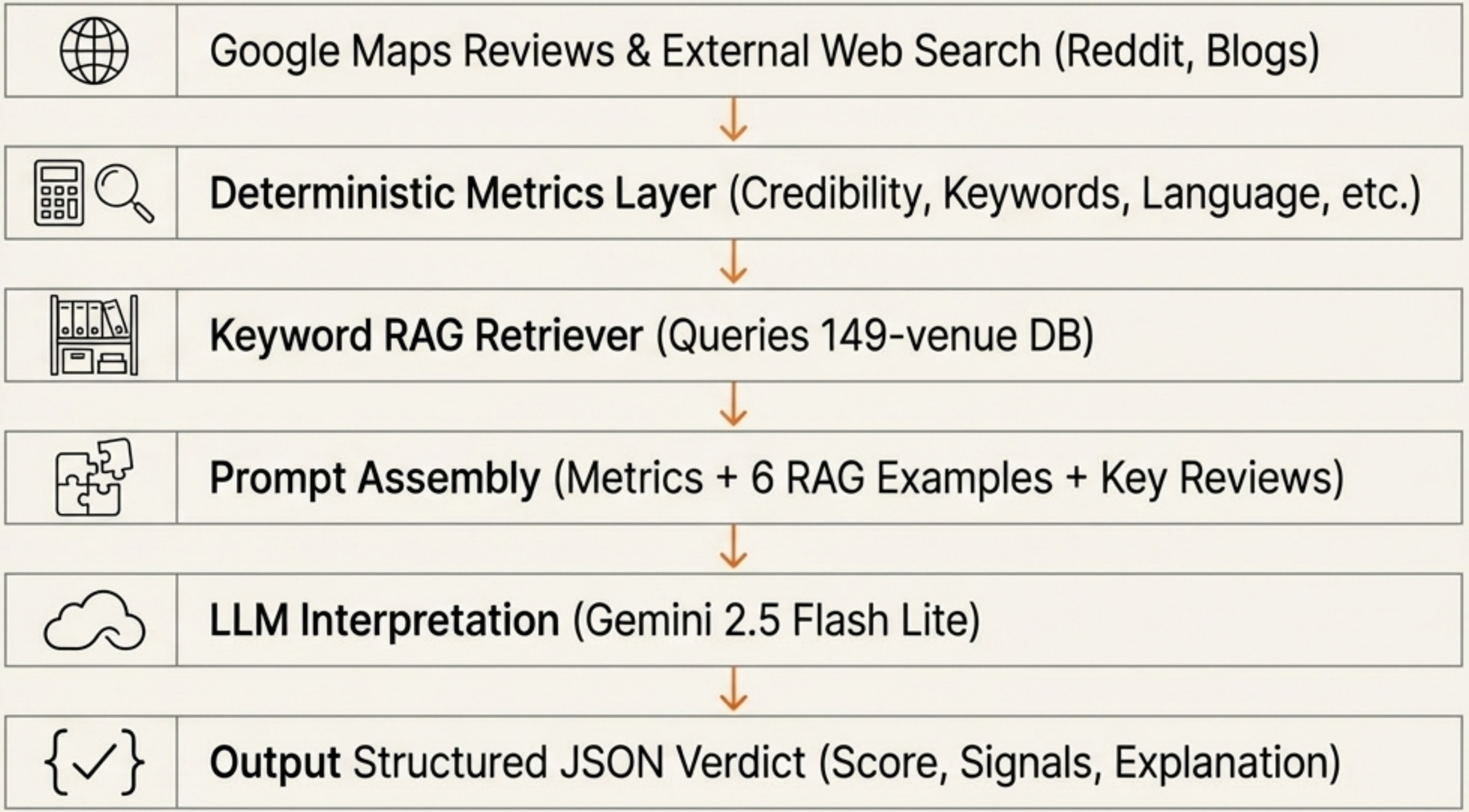
Before making a final judgment, the model now retrieves the 6 most similar venues from this database to calibrate its score.

We pitted two RAG approaches against each other.

Metric	Baseline	Keyword RAG	Vector RAG
MAE (v1)	21.25	**17.45**	18.05
Latency	3.2s	**3.3s (+3%)**	4.1s (+26%)
Dependencies	None	**None (Pure Python)**	ChromaDB

Keyword RAG delivered **better accuracy** with minimal overhead and no heavy dependencies, making it the clear choice for this project's scale.

This is the final TrapCheck engine.



To truly test the system, we built a much tougher evaluation.

Stratified Test Set

30 venues sampled
from RAG DB

Realistic Synthetic Data

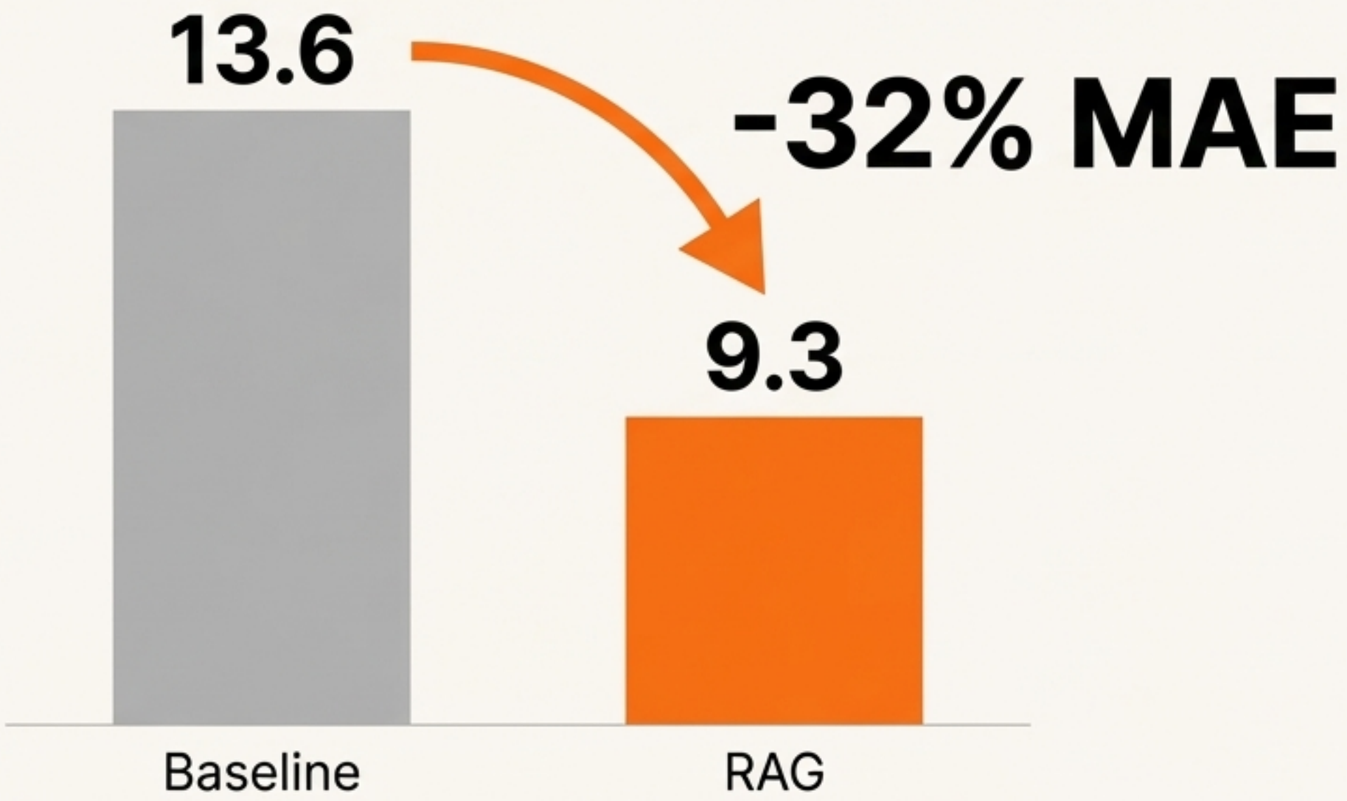
We injected noise: contradictory reviews, ambiguous 3-star ratings, and varied review counts (8-12)

Rigorous Method

Leave-one-out validation with 3 runs per venue
(90 total runs per experiment)

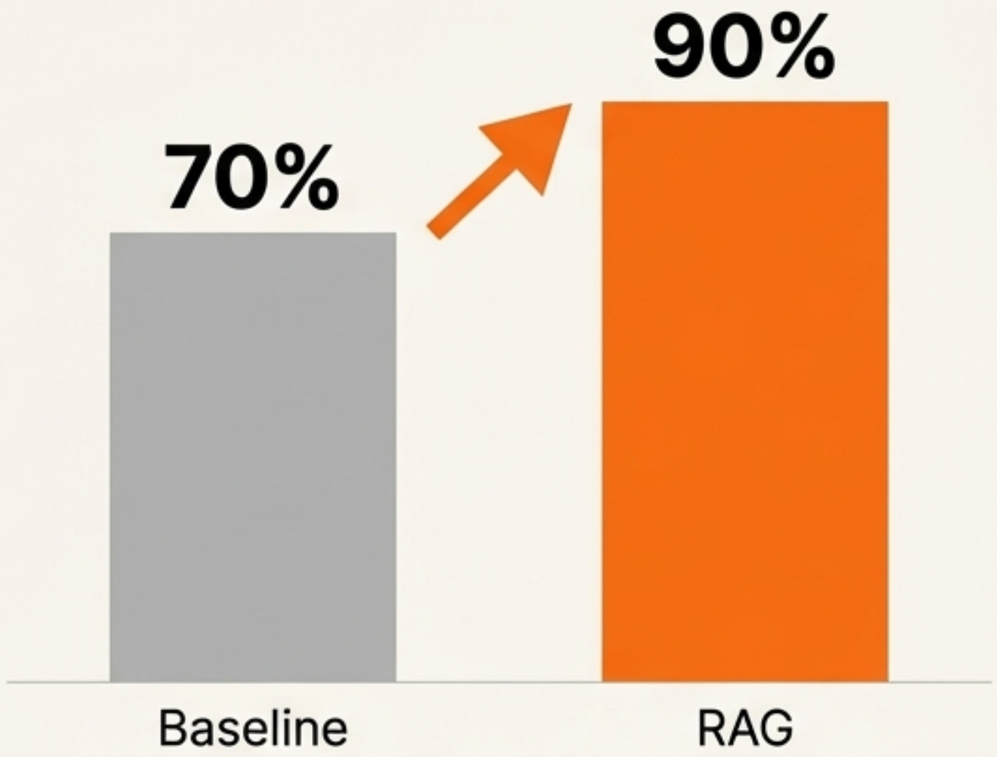
This prevents the model from “cheating” by pattern-matching on overly clean, predictable data.

The final system delivered a 32% reduction in error.



Overall category accuracy increased from 90.0% to 95.6%.

Most importantly, RAG conquered the ambiguous “Mixed Zone.”



+20% Accuracy Boost
where it mattered most.

RAG provides the crucial calibration for ambiguous cases where the metrics layer alone is not enough. This was the largest single improvement in the entire project.

The key lesson: separate deterministic tasks from interpretive ones.

Let Deterministic Code...

...count.

...match.

...and score.

...Let the LLM...

...interpret.

...reason.

...and explain.

Limitations & Future Work

Persistent edge cases (e.g., Harry's Bar Venice) remain a challenge.
The RAG database is English-centric and underrepresents some venue types like "tours."
Next step: Scale the RAG DB to 500+ examples and switch to Vector RAG.