From Reactive Prompts to a Proactive Knowledge Runtime

Evolving Perplexity's AI Agents to Select, Not Just Generate

We propose evolving our AI agents to a Faceted Knowledge Runtime.

This architectural shift moves agent intelligence from brittle, monolithic XML prompts into a dynamic, queryable knowledge base.

Agents can select the best-proven action from thousands of scenarios in 50-150ms, *before* **they generate a response.**

For Support

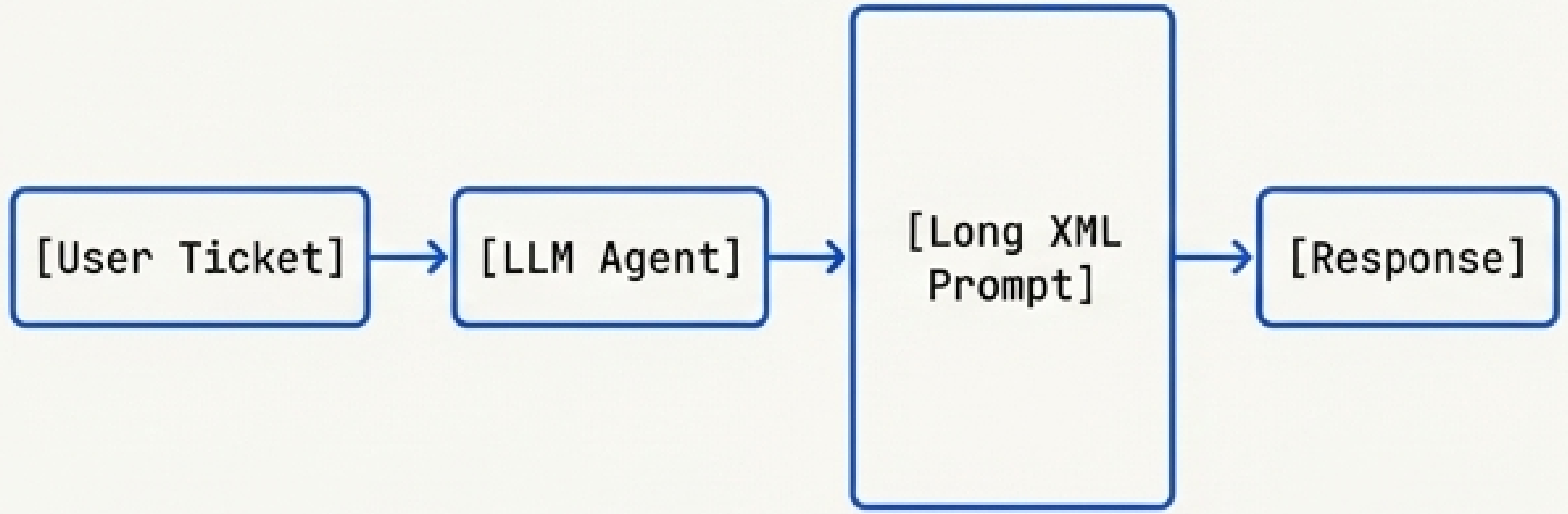
A step-change in resolution speed, scale, and governance.

For Product

Unlocks proactive, in-product user engagement and self-improving systems.

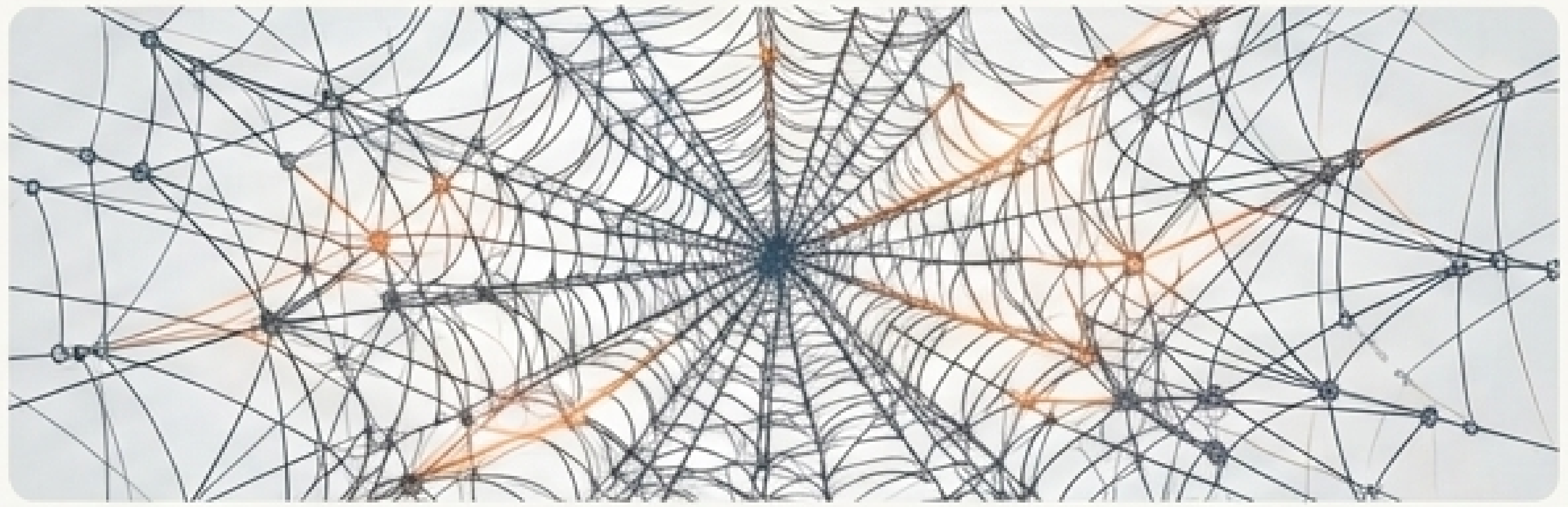
Our current XML-based agents have been successful, but are reaching their scaling limits.

The Current Model



This model got us here, proving the value of AI-driven support.

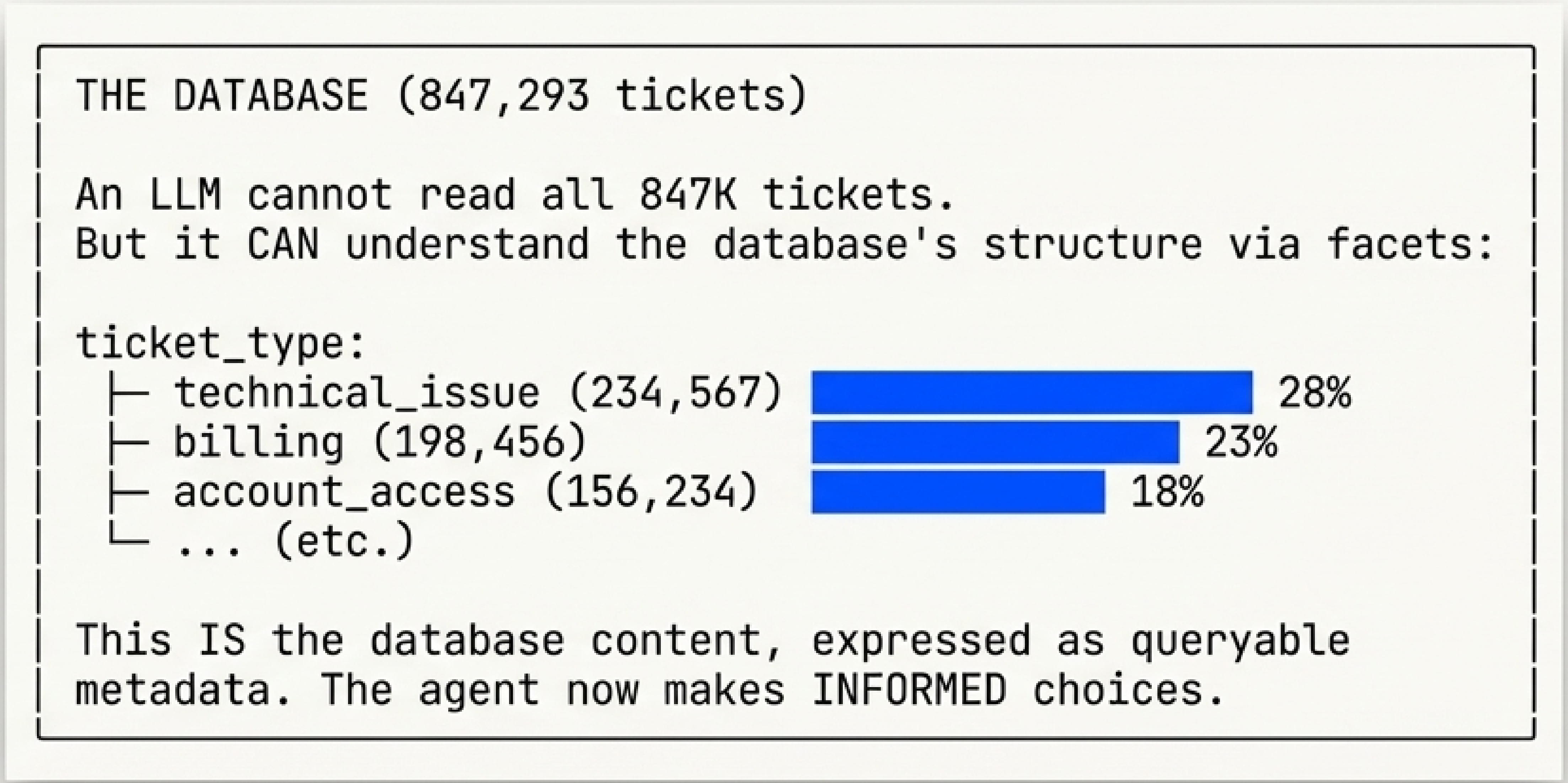
The Underlying Reality



Key Problems

- **Brittle:** A small change in one rule can have cascading, unpredictable effects.
- **High Maintenance:** Adding a new scenario requires careful, time-consuming XML engineering and testing.
- **Unscalable:** The prompt length and complexity grow linearly with each new scenario, hitting context window limits.
- **Cannot Learn:** The system has no mechanism to identify and prioritize which rules perform best; it only executes.

The foundation of the new runtime is Facets: a metadata language for our entire support database.

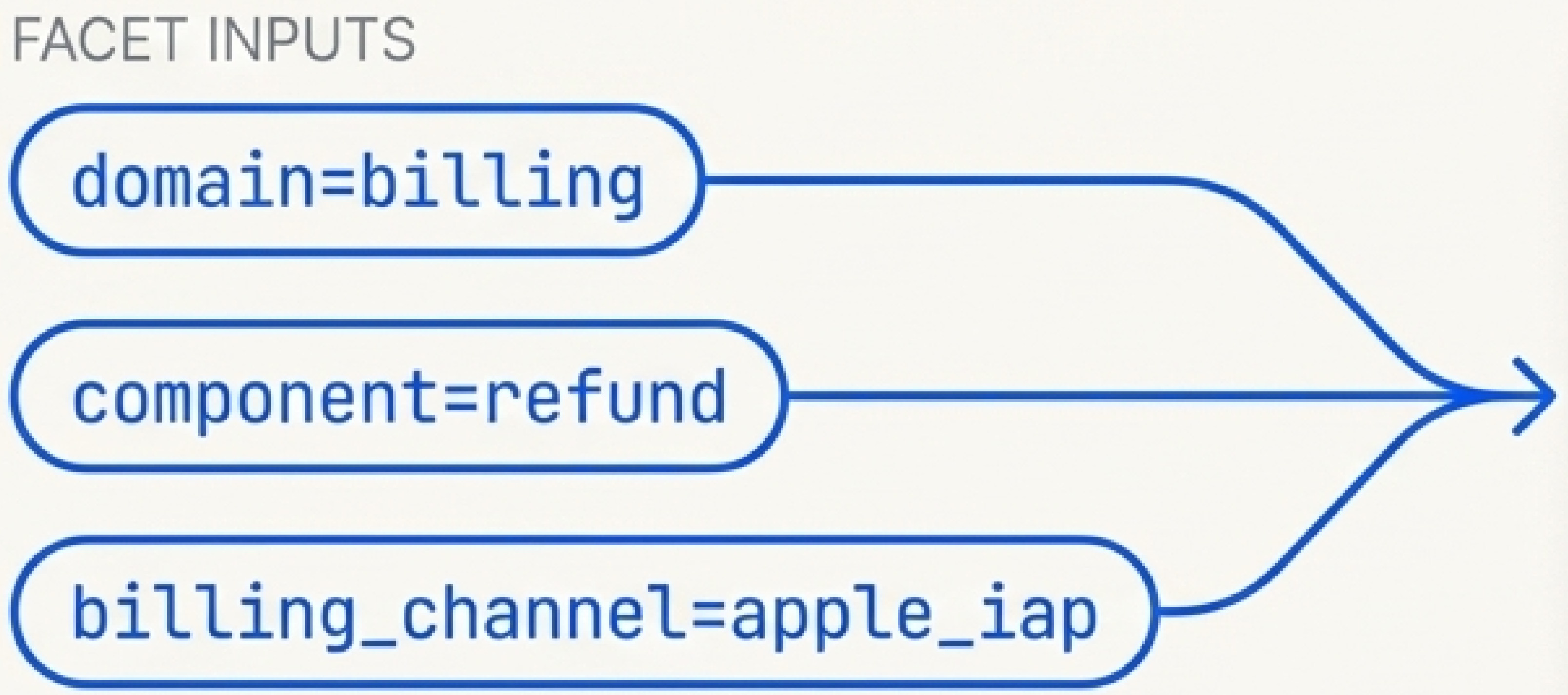


```
THE DATABASE (847,293 tickets)

An LLM cannot read all 847K tickets.
But it CAN understand the database's structure via facets:

ticket_type:
 ├─ technical_issue (234,567)                    28%
 ├─ billing (198,456)                        23%
 ├─ account_access (156,234)             18%
 └─ ... (etc.)

This IS the database content, expressed as queryable
metadata. The agent now makes INFORMED choices.
```

Instead of feeding the LLM raw text, we provide it a structured, hierarchical understanding of our entire knowledge space. The agent can now reason over distributions and correlations to find the most relevant context, just like faceted search.

Playbooks are version-controlled, performance-scored recipes for resolution.

If Facets describe *what* a ticket is about, Playbooks define *how* to solve it. They are discrete, executable objects stored in our database, not embedded in a prompt.



PLAYBOOK: BILL-REFUND-APPLE-111

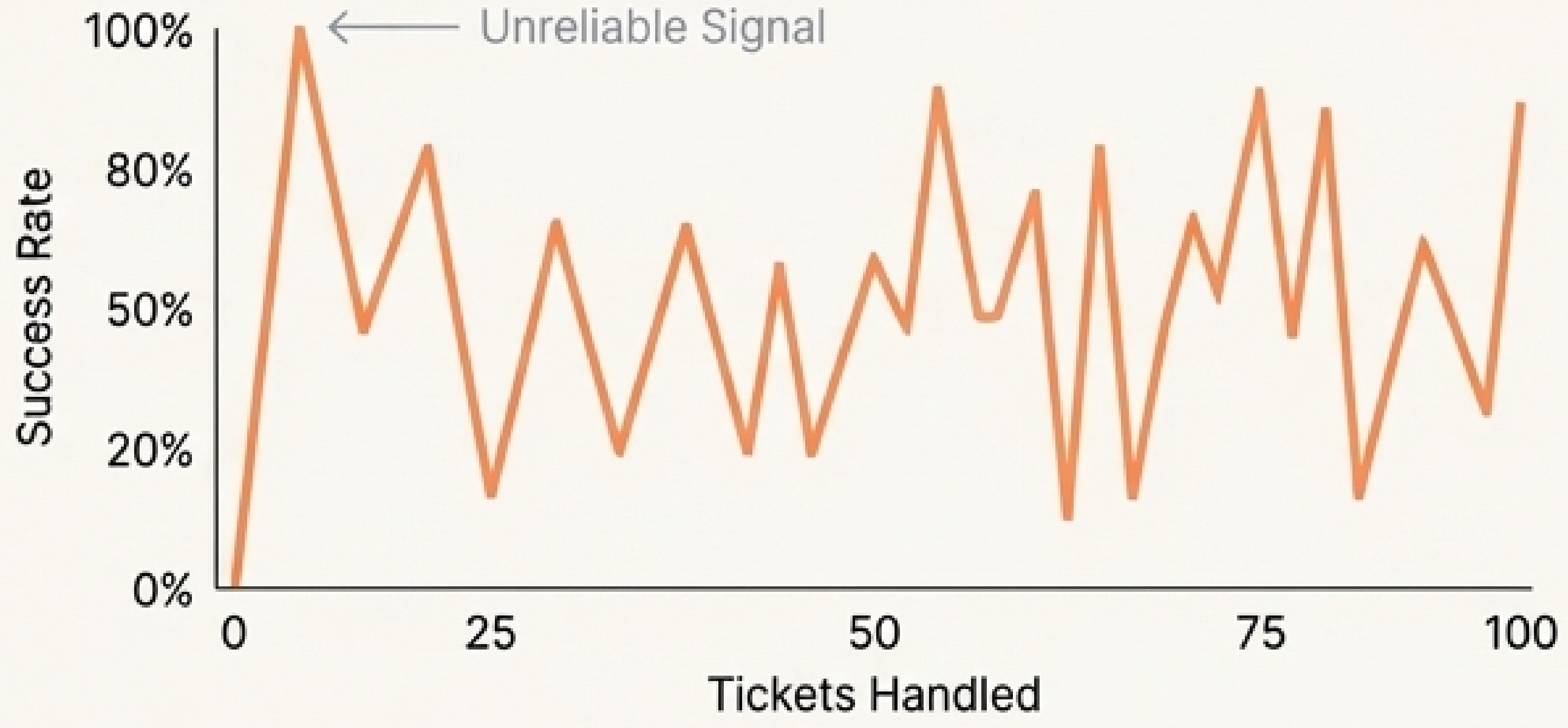
Playbook Name	Refund Request / Apple IAP
Match Clauses	`domain=billing,` `billing_channel=apple_iap`
Steps	1. (Tool Call) `` `lookup_subscription(user_id)` `` 2. (Policy Check) Explain that Apple manages IAP refunds directly. 3. (Response Template) Provide link and instructions for Apple's refund process.
Risk Ceiling	`low`
Performance Score	**0.92** (success_rate_smoothed)

The system self-improves by scoring Playbooks on performance, with statistical smoothing.

We don't just execute Playbooks; we measure their outcomes (success rate, resolution time, CSAT) and use that data to rank them for future tickets.

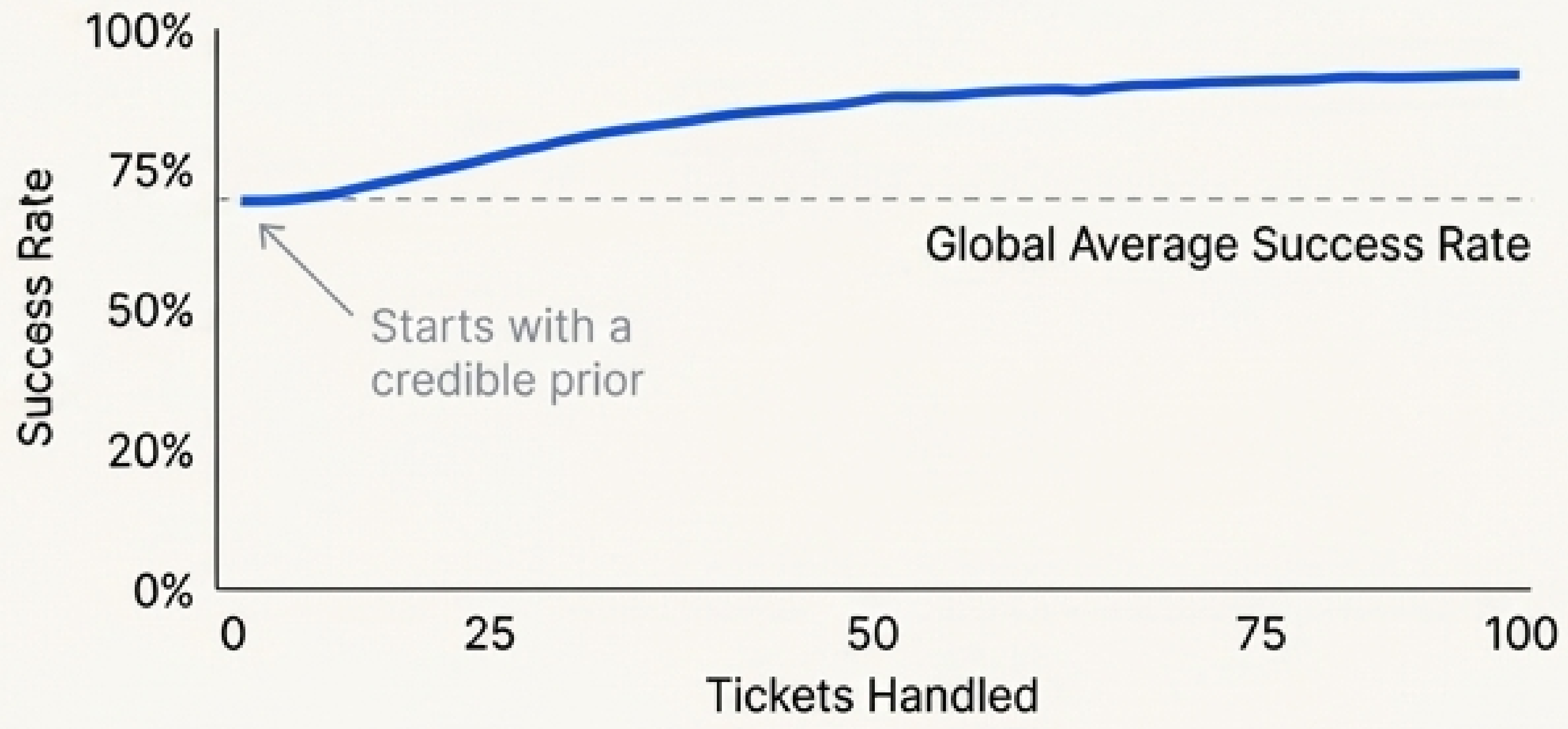
Without Smoothing

A new Playbook with one lucky success gets a 100% score, unfairly promoting it.



With Smoothing

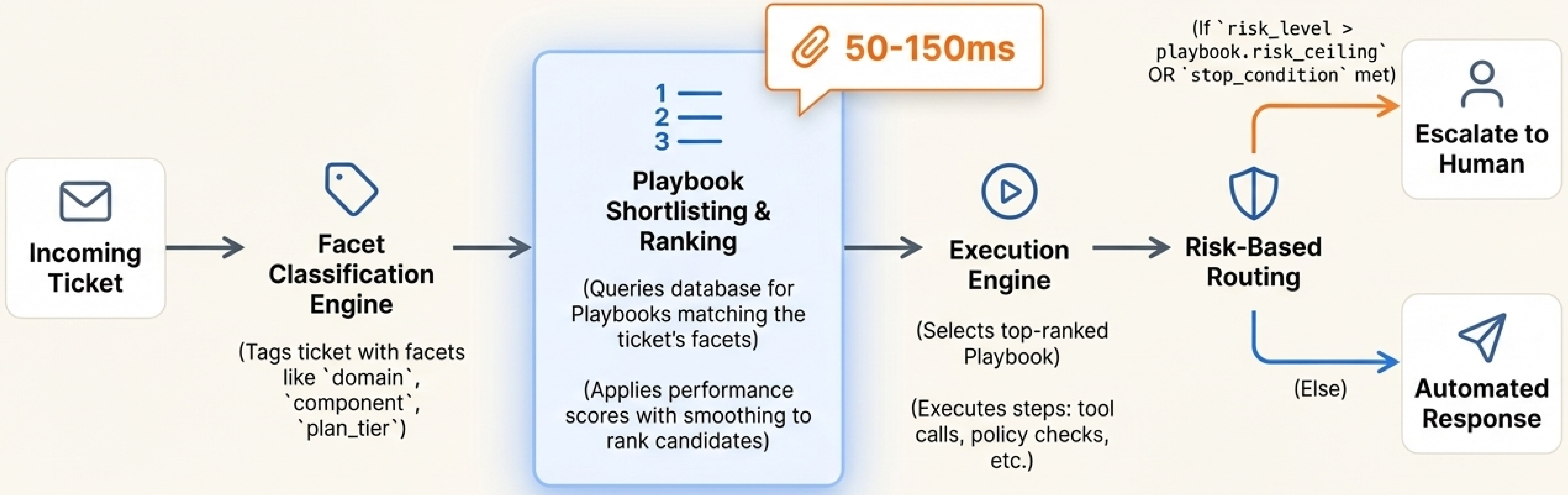
Bayesian smoothing (Beta-Binomial) pulls the score of new or low-volume Playbooks towards the global average. This "shrinks" outlier results, ensuring only consistently high-performing Playbooks rise to the top.



This mechanism prevents volatility and ensures the agent reliably selects paths that are proven to work at scale.

NotebookLM

The Faceted Knowledge Runtime: A high-level view.


Incoming Ticket
Facet Classification Engine
(Tags ticket with facets like `domain`, `component`, `plan_tier`)
50-150ms
Playbook Shortlisting & Ranking
(Queries database for Playbooks matching the ticket's facets)
(Applies performance scores with smoothing to rank candidates)
Execution Engine
(Selects top-ranked Playbook)
(Executes steps: tool calls, policy checks, etc.)
Risk-Based Routing
(If `risk_level > playbook.risk_ceiling` OR `stop_condition` met)
Escalate to Human
(Else)
Automated Response

The system's logic is governed by a central Facet Registry.

The registry is the “source of truth” that defines every facet, its hierarchy, its data source, and its relationship to other entities. This ensures consistency and allows the agent to reason about the data model itself.

```
{
    "facet_name": "component",
    "description": "Granular component inside domain",
    "kind": "categorical",
    "allowed_values_by_parent": {
      "parent_facet": "domain",
      "map": {
         "billing": [ "payment_failed", "refund", ... ],
         "account_access": [ "google_signin", ... ]
      }
    },
    "sources": [{ "type": "classifier", "from": "ticket.text" }],
    ...
},
{
    "facet_name": "risk_level",
    "description": "Computed risk for escalation decisions",
    "kind": "categorical",
    "sources": [{
        "type": "computed",
        "expression": "risk_model_v1(...)",
        "depends_on": [ "domain", "customer_segment", ... ]
    }]
}
```

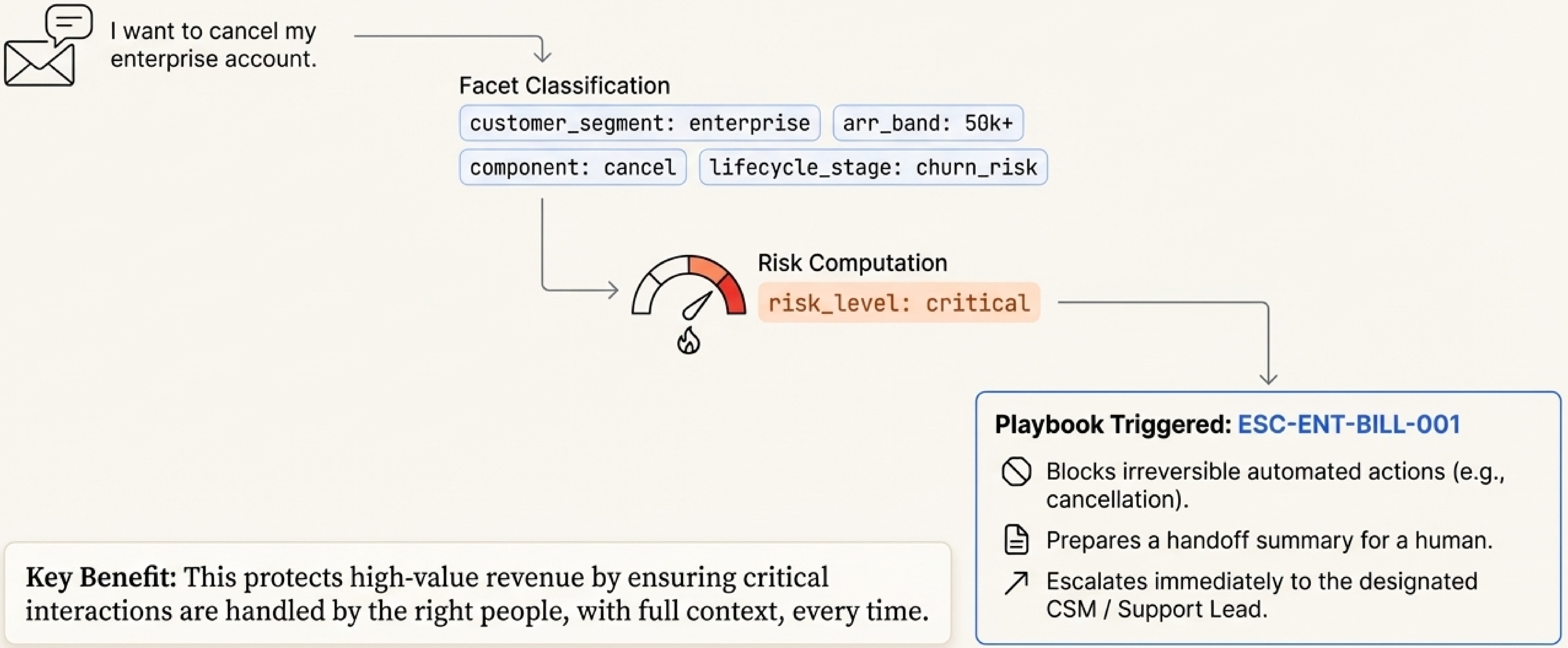
- **Definitions & Hierarchies:** Defines parent-child relationships (e.g., `domain` -> `component`).
- **Source Mapping:** Specifies where each facet’s value comes from (e.g., a classifier, a DB join, or a computed rule).
- **Join Policy:** Defines which data entities can be connected, ensuring security and performance.
- **Exploration Policy:** Guides the agent on which facets provide the most information gain.

The runtime delivers quantifiable gains in speed, scale, and quality.

Dimension	Current State (XML Prompts)	Faceted Knowledge Runtime
Speed (TTR)	LLM must parse long prompt context before acting.	Agent selects a proven, optimized Playbook **instantly.**
Scalability	High cost: Add 1 scenario = engineer & test complex XML logic.	**Low cost:** Add 1 scenario = insert a new Playbook object into a DB.
Maintenance	High overhead. Prompts are monolithic and difficult to debug.	**Low overhead.** Playbooks are modular, versioned, and individually testable.
Quality	Static. Quality depends on initial prompt engineering.	**Self-Improving.** Performance smoothing automatically promotes effective Playbooks.
Risk Mgmt.	Implicit. Risk rules are buried in complex conditional logic.	**Explicit.** Risk is a computed facet (`risk_level`) used for routing.

Explicit risk modeling enables intelligent, revenue-aware governance

Instead of treating all tickets equally, the runtime computes a `risk_level` for each interaction based on a combination of facets.


I want to cancel my enterprise account.
Facet Classification
customer_segment: enterprise
arr_band: 50k+
component: cancel
lifecycle_stage: churn_risk
Risk Computation
risk_level: critical

Playbook Triggered: ESC-ENT-BILL-001

- Blocks irreversible automated actions (e.g., cancellation).
- Prepares a handoff summary for a human.
- Escalates immediately to the designated CSM / Support Lead.

Key Benefit: This protects high-value revenue by ensuring critical interactions are handled by the right people, with full context, every time.

This architecture is the key to unlocking the next strategic horizon for AI at Perplexity.

From **Reactive Agents** that *answer questions*…

↓

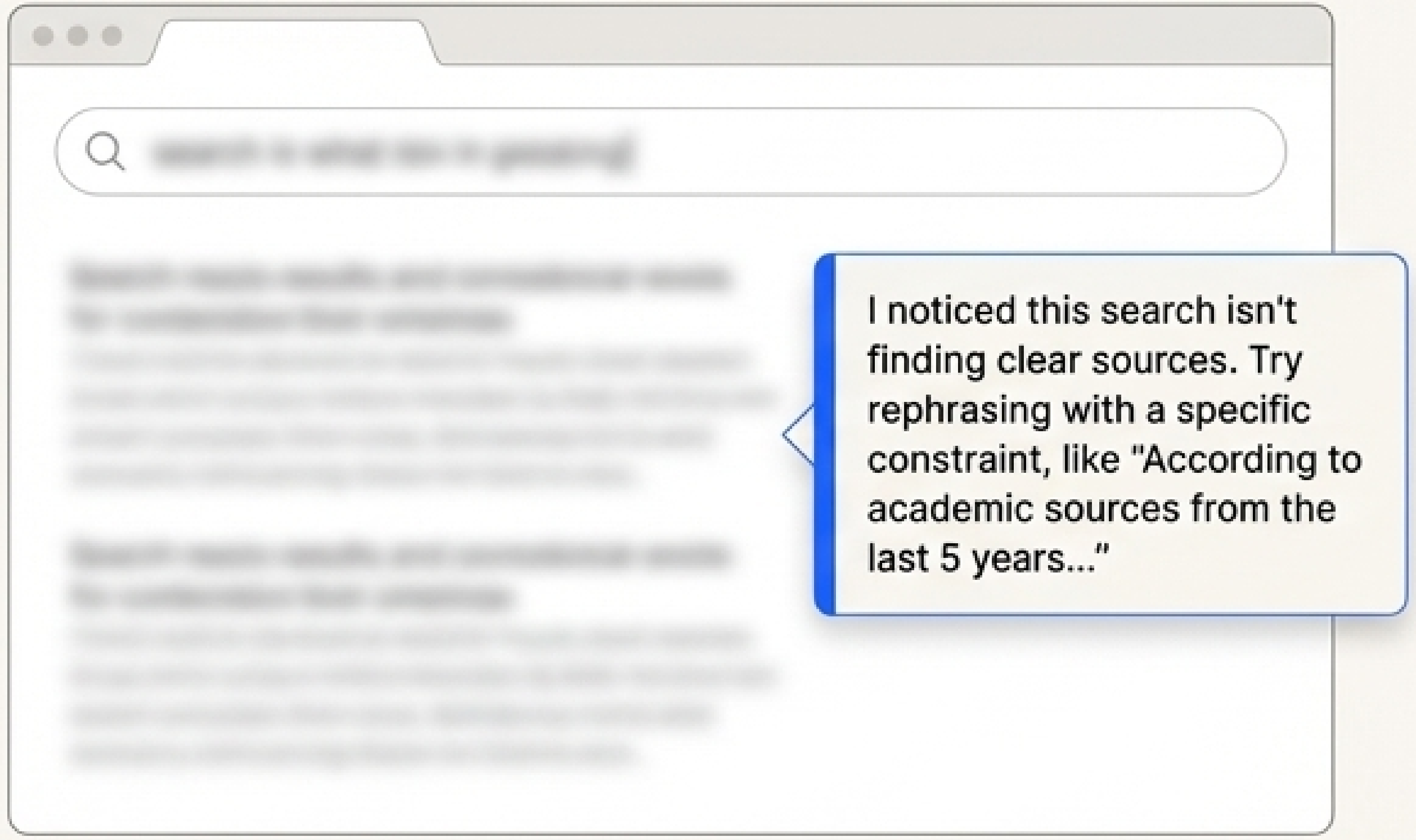
…to **Proactive Agents** that *prevent questions*.

With a deep, faceted understanding of the user (`lifecycle_stage`, `usage_band`) and their context (`component`, `symptom`), our agents can move beyond the support queue and become an integrated part of the product experience. They can anticipate needs, guide users, and actively improve engagement.

Imagine proactive agents embedded directly within Comet and the core product.

Proactive Prompt Engineering Tip

Source Serif Pro

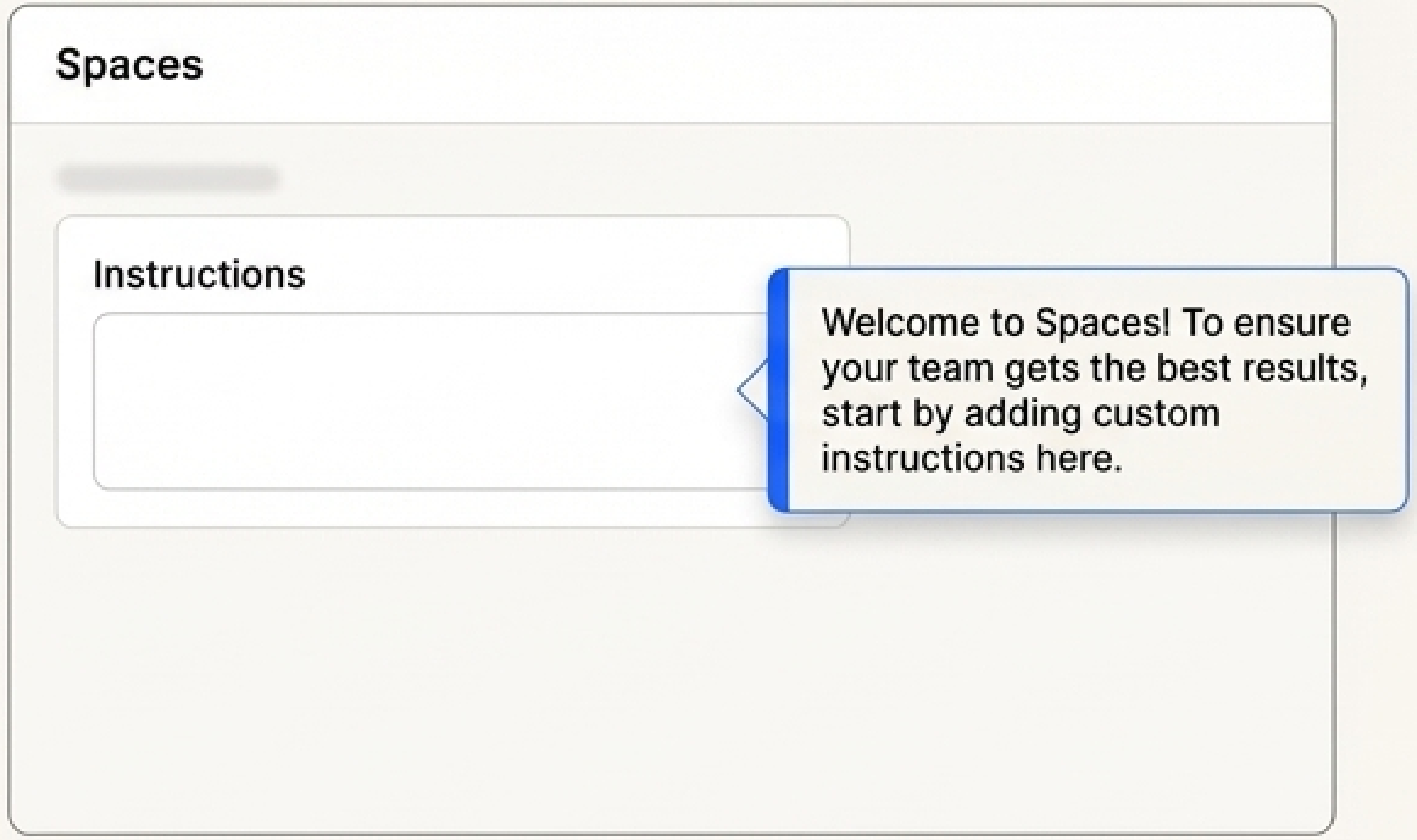


I noticed this search isn't finding clear sources. Try rephrasing with a specific constraint, like "According to academic sources from the last 5 years..."

Triggered by (domain=content_quality, symptom=poor_sources, usage_band=high)

Intelligent Onboarding

Source Serif Pro



Welcome to Spaces! To ensure your team gets the best results, start by adding custom instructions here.

Triggered by (lifecycle_stage=onboarding, component=spaces, user_role=org_admin)

We can achieve this vision through a phased, iterative rollout.

Phase 1: Minimum Viable Facets (MVF) & Core Routing

Goal: Prove the core architecture and handle top-volume tickets.

- Implement 6 core facet dimensions: `request_type`, `domain`, `component`, `plan_tier`, `customer_segment`, `billing_channel`.
- Define 15-25 Playbooks for the highest volume and highest risk scenarios.
- Build simple, rules-based escalation triggers.

Outcome: A robust foundation for routing and resolving the majority of tickets, with safer escalation.

Phase 2: Advanced Intelligence & Proactive Engagement

Goal: Activate the self-improving loop and begin in-product pilots.

- Add advanced context facets: `lifecycle_stage`, `usage_band`, `arr_band`, `symptom`.
- Fully activate performance smoothing for Playbook ranking.
- Develop and pilot the first proactive agent interventions inside Comet.

Outcome: A self-improving system that actively enhances the product experience.

This is not an upgrade to our support system.

It is the foundation for an in-product intelligence layer that understands user context and actively improves their experience.