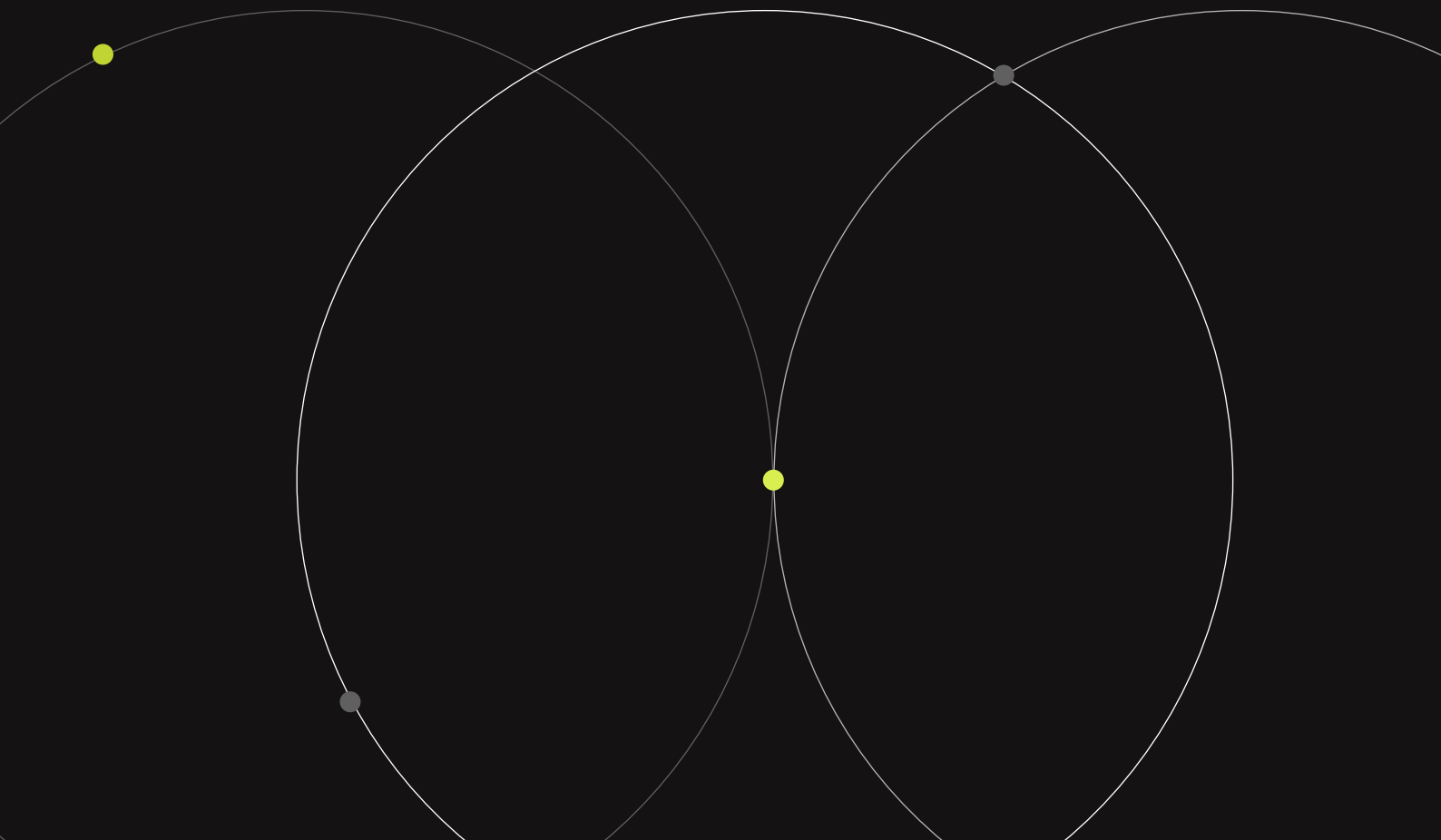


WHITEPAPER

# Raven 3.5

The purpose-built LLM for  
enterprise customer service.



# Contents

---

The AI call center production reality .....	3
The architectural mismatch .....	4
The Raven training method.....	6
The need for speed .....	8
Multilingual capabilities .....	10
As the Raven flies: The roadmap .....	12
Decision framework: Raven vs other models.....	13



Consider a hotel guest calling to modify their reservation, a simple request that should take under a minute.

Guest

Hi, I need to change my checkout date.

[2.5 second pause]

Hello? Is anyone there?

Agent

I'd be delighted to assist you with modifying your reservation today! To help you with this request, I'll need to look up your current booking in our system. Could you please provide me with your confirmation number, last name, or the email address you used when making the reservation?

It's under Chen, confirmation 8847.

[2 second pause]

Thank you for providing that information! I've found your reservation—you're currently scheduled to check out on March 15th. What date would you like to change it to?

I need to leave a day early, the 14th. Oh, and is breakfast included?

[2.3 second pause]

I can help you with both of those questions! First, regarding your checkout date change to March 14th, let me process that modification for you. As for breakfast—yes, we offer a complimentary continental breakfast for all guests. It's served daily from 6:30 AM to 10:00 AM in our main dining area on the lobby level. Would you like me to provide you with more details about what's included—

Stop, stop. I just need to know the new total.

[2.1 second pause]

[API error: Early checkout requires manager approval]

Based on our standard rates, your new total should be approximately \$347. However, I should mention that in some cases, modifications to reservations may be subject to—

[Hangs up]

Total time	~120 seconds Customer abandoned. Modification never processed. Agent hallucinated pricing.
------------	---

Guest

Hi, I need to change my checkout date.

[0.4 second pause]

Agent

Sure, what's your confirmation number?

It's under Chen, confirmation 8847.

[0.5 second pause]

Found it. You're checking out March 15th. What day do you need?

The 14th. Is breakfast included?

[0.4 second pause]

Yes, continental breakfast from 6:30 to 10—

Great. What's the new total?

Let me check that. [0.6 second pause]

The 14th needs manager approval for your rate. I can submit the request now. You'll get an email within an hour with the final total.

Yes please.

Done. Check your email in about an hour.

Total time	~45 seconds Request submitted. Customer satisfied.
------------	---



# The architectural mismatch: Generalist vs. Specialist LLMs

General-purpose LLMs like ChatGPT and Claude are good at broad capabilities across text-based tasks, but this creates foundational bottlenecks for live customer service.

PolyAI built Raven to solve a specific technical problem: achieving sub-300ms inference latency while maintaining strict, domain-specific accuracy through specialization rather than scale. In fact, Raven 3.5 outperforms GPT-5 and Claude Sonnet 4.6 across all four of our customer service benchmarks: English LLM score, Multilingual score, Style, and Date-time logic.

## 1. Model size vs. inference latency

The core tradeoff in AI is simple: larger models deliver broader capabilities but slower inference. Conversational naturalness requires sub-300ms turnaround times—a hard constraint that massive models on shared API infrastructure cannot reliably meet.

- **Right-sized architecture:** OpenAI optimizes for capability breadth, requiring enormous parameter counts. Raven uses the smallest, most efficient model size required to handle the domain, running on dedicated infrastructure to guarantee speed under load.
- **Latency-optimized auto-reasoning:** Raven dynamically decides when to deliberate deeply and when to respond directly. By producing fewer than 40 tokens of internal reasoning in milliseconds, it handles complex requests flawlessly without adding latency to simple conversational turns.
- **Omnichannel flexibility:** Raven 3.5 provides native webchat support, working across both voice and text channels from a single model. This makes it straightforward to deploy consistent, low-latency AI across your contact center.

## 2. Training distribution mismatch

Models trained primarily on internet text exhibit systematic behavioral biases that create immediate failures in live production environments:

- **Useless formatting vs. emotion tags:** Text models generate markdown (bullet points, code blocks) that add no value to a voice call and waste inference cycles. Raven replaces this with emotion tags (e.g., [apologetic], [friendly]) that instruct compatible text-to-speech models on exactly how to deliver the response.
- **Information density:** Text allows users to skim; voice delivers information at a fixed rate (~150 words/minute). General models default to verbose monologues, whereas Raven is calibrated to deliver concise information at a natural spoken pace.
- **Hallucination under uncertainty:** Text models are heavily biased toward always providing an answer. Raven is explicitly trained to recognize when relevant information isn't available, responding with "I don't know" or safely escalating rather than hallucinating a policy.

## 3. Function-calling reliability

Production agents orchestrate API calls, database lookups, and multi-step workflows in real time. General LLMs optimize their function calling for coding assistants. Raven's function calling is trained specifically for the messy realities of enterprise APIs:

- **Graceful delays:** Raven naturally generates appropriate conversational filler when waiting on external backend lookups.



- **Real-time error handling:** When APIs time out or return errors, Raven accurately surfaces those constraints rather than hallucinating alternate workflows.
- **Multi-step orchestration:** Raven successfully manages complex sequences and conditional logic without losing the conversational context.

## Quantifying production impact

These individual failures compound exponentially across a multi-turn conversation. Consider a standard 20-turn customer service call:

- **Compounding accuracy:** A 90% per-turn accuracy yields just a 12% overall conversation success rate ( $0.90^{20}$ ). A 98% per-turn accuracy yields 67% ( $0.98^{20}$ ).
- **Compounding latency:** 20 two-second processing pauses result in 40 seconds of frustrating dead air.

## The specialization approach

The key technical insight is this: for production customer service deployments, broad general capability is unnecessary.

Raven eliminates the speed/accuracy tradeoff by focusing all model capacity on the target use case. By training exclusively on millions of real, anonymized customer service conversations, Raven achieves the exact behavioral defaults enterprise contact centers need—without relying on fragile prompt engineering.



# The Raven training methodology

Raven isn't trained from scratch. It's based on open-weights foundation models that have undergone large-scale pre-training. PolyAI applies a three-stage post-training process to transform these into specialized voice agents.

- STAGE 1:

## Supervised fine-tuning (SFT)

**Objective:**

Distilling high-level reasoning capabilities into a latency-optimized voice model.

**The methodology**

SFT trains using gold labels, where massive, reasoning-based LLMs analyze real customer conversations to generate ideal responses. While these reasoning models are too slow for live voice interactions, SFT allows us to distill their accuracy and intelligence into our faster, proprietary model.

**Training data:**

Hundreds of thousands of real production conversations from PolyAI deployments across restaurants, healthcare, banking, and telecommunications.

**What SFT accomplishes:**

By training on these high-quality, curated examples, the model learns voice-specific norms that general LLMs lack:

- **Information Density:** Concise delivery suitable for spoken audio rather than reading.
- **Operational Logic:** Knowing when to call functions/APIs versus responding directly.
- **Safety:** Identifying when to say "I don't know" rather than hallucinating.

**Key takeaway:**

SFT creates a model with the sophisticated reasoning of a massive LLM but the speed required for natural, real-time conversation—eliminating the need for complex prompt engineering.



- **STAGE 2:**

## Continuous optimization (GRPO + DPO)

A major breakthrough in Raven 3.5's post-training recipe is the purposeful fusion of two advanced optimization techniques: GRPO and DPO. Rather than treating continuous optimization and preference tuning as isolated concepts, our pipeline weaves them into a single, iterative training engine.

To understand why this combined approach is so powerful, it helps to understand how each technique works on its own:

**GRPO (Generalized Reinforcement Policy Optimization):** This is essentially active practice. The model is given a scenario, generates multiple possible responses, and an automated reward system scores them. Over time, the model hill-climbs, adjusting its internal weights to favor the types of responses that yield the highest scores. It is fantastic for general refinement, but it has a blind spot: if the model is fundamentally confused and none of its generated responses are good, GRPO has no positive behavior to reinforce.

**DPO (Direct Preference Optimization):** This is contrastive learning. Instead of letting the model practice freely, we forcefully present it with a pair of responses: one poor (what the model tried to say) and one gold-standard (what an expert would say). DPO mathematically penalizes the bad response and reinforces the good one. It teaches the model exact boundaries by explicitly showing it "do this, not that."

**Why the fusion matters:** By fusing these techniques together in Raven 3.5, we cover their individual weaknesses. We allow GRPO to run continuously so the model can practice and naturally optimize its conversational flow. However, the moment the model hits a wall or consistently fails at a specific type of turn—creating a gap where GRPO has nothing to reward—our system injects a targeted DPO pair. This immediately shows the model exactly what "good" looks like, unblocking the GRPO process and allowing it to continue optimizing safely and accurately.

## The compounding effect

**These stages compound:** SFT establishes voice-appropriate defaults, preference tuning refines quality within those defaults, and GRPO scales improvement through production data. Every parameter update focuses exclusively on customer service voice conversations.

You wouldn't (necessarily) care if your human CSR can juggle knives, compose French verse, and invent on-the-spot recipes from the dregs of your refrigerator. Likewise, Raven isn't learning to be better at everything, just customer service.

### The data advantage

Raven's ~90% deployment concentration creates a virtuous data flywheel. Production conversations across diverse industries (different telephony infrastructure, backend systems, load patterns) surface real edge cases and provide volume to train domain-specific behaviors without overfitting. General-purpose model providers can't replicate this domain-specific feedback loop.



# The need for speed

Voice requires <500ms turn-around time—a hard constraint. Raven achieves consistent <300ms latency through three architectural decisions:

## 1. Model size optimization

General-purpose LLMs require massive parameter counts to excel at everything from code generation to creative writing. Raven uses the most efficient model size that reliably handles enterprise customer service, generating tokens significantly faster than bloated generalist models.

Furthermore, Raven 3.5 introduces latency-optimized auto-reasoning. The model dynamically decides when a conversational turn requires deeper deliberation and when it can respond directly. By producing fewer than 40 tokens of internal reasoning in just a few hundred milliseconds, it flawlessly processes complex requests without adding noticeable latency to simple turns.

## 2. Self-hosted dedicated infrastructure

Proprietary API-based models run on shared cloud infrastructure where response times vary wildly based on global load, creating unpredictable tail latencies. Raven runs on dedicated infrastructure optimized for our exact deployment patterns:

- **Consistent performance:** No competing for compute resources. Response times remain stable under heavy call volume.
- **Optimized for voice workloads:** Infrastructure is specifically tuned for customer service conversation patterns—typical turn lengths, function-calling frequency, and retrieval integrations.
- **Elimination of network overhead:** Self-hosted deployment reduces round-trip time compared to external API calls.

## 3. Optimizing output format

Generalist models are primarily trained on visual text, so they naturally default to formatting like bullet points, numbered lists, and markdown. In a voice environment, these tokens add no value, waste inference cycles, and sound robotic when read aloud by a Text-to-Speech (TTS) engine.

Raven 3.5 solves this through two distinct, voice-first capabilities:

- **Natively conversational output:** Because Raven is trained on hundreds of thousands of real phone calls, its baseline behavior is natively conversational. It generates responses exactly as they are meant to be spoken, completely eliminating the latency and awkwardness of markdown overhead.
- **Emotion tags for TTS:** While it naturally strips out visual formatting, Raven 3.5 can be explicitly prompted to include emotional context tags within its output. By generating tags such as [friendly], [compassionate], or [apologetic], Raven gives compatible TTS engines the exact instructions they need to match the tone to the moment. Additionally, the model gracefully handles backend API delays by generating natural conversational filler, disguising backend latency from the caller without relying on stiff, scripted prompts.



## More time takes more time

Individual latency improvements multiply exponentially across multi-turn conversations. On a typical 20-turn customer service call:

- **General-purpose LLM:** (~2 seconds per turn) = 40 seconds of dead air
- **Raven 3.5:** (sub-300ms per turn) = seamless, natural conversational flow.

This speed directly drives operational efficiency. Faster conversations mean higher throughput per line, lower telephony costs, and more completed transactions per hour.

## The speed ecosystem & the road to Omni

Speed optimization extends beyond the LLM. PolyAI's current production system utilizes a tightly orchestrated stack of models: Speech Recognition (ASR), Retrieval, the LLM (Raven 3.5), and Text-to-Speech (TTS). For example, our retrieval model deliberately over-retrieves data at high speeds, relying on Raven's training to instantly ignore irrelevant context rather than bottlenecking the system with slow, high-precision retrieval.

**Looking ahead:** The next evolution in reducing latency is eliminating the hops between these models entirely. Next on our roadmap is Raven Omni, an audio-native model that fuses speech recognition directly with the LLM. Instead of piping audio through a separate ASR system and handing text to the language model, Raven Omni will apply the reasoning power of the LLM directly to speech understanding—hearing what the caller said and how they said it, then responding directly.



# Multilingual capabilities

Raven 3.5 supports 23 languages with perfect language adherence, meaning enterprise teams can write their system prompts and source documents in English, and ask the agent to respond to the caller in any other supported language. This addresses a critical production constraint for global enterprises: deploying unified voice and chat agents across regions without quality degradation or the need for separate, language-specific configurations.

## Multilingual is a specialized skill

General-purpose models can speak many languages, but they exhibit systematic problems when pushed into production environments:

- **Variable quality across languages.** Because generalist training data heavily skews toward English, high-resource languages perform reasonably well, while lower-resource languages suffer from degraded accuracy, unnatural phrasing, and higher hallucination rates.
- **Unreliable language switching.** General models often slip into other languages mid-conversation. This is particularly problematic when the agent's instructions are written in English, but it needs to speak to the customer in another language.
- **Inconsistent function calling across languages.** Public models tend to reason in their most comfortable language (usually English) and translate their response on the fly. This creates noticeable latency and often results in unnatural, robotic phrasing.

## Raven 3.5's polyglot architecture

Raven 3.5 delivers a massive step forward in multilingual naturalness and performance. It addresses the fundamental tensions of multilingual AI through several specialized architectural decisions:

- **Native scratchpad:** Raven 3.5 completely avoids the translation lag of generalist models. It uses its hidden reasoning trace as a target-language "scratchpad," allowing the model to actually think in the target language (like Spanish) before speaking in that language.

- **Perfect language adherence:** The model never accidentally switches languages mid-conversation unless explicitly instructed. This is an architectural guarantee, not a probabilistic guess.
- **English prompting, native output:** Agent configurations written in English can be deployed globally, with the agent speaking the appropriate local language. Raven separates the configuration language from the output language natively, requiring no slow, explicit translation steps.
- **Language-specific behavioral norms:** Raven 3.5 is explicitly trained to respect the cultural and behavioral norms of its 23 supported languages. For example, the model is actively penalized for using ambiguous gender terms in languages like Spanish or Polish, where getting it wrong sounds immediately unnatural to a native speaker.

## Operational simplification

These capabilities dramatically simplify enterprise operations. A single agent configuration can be deployed to multiple countries. There is no need for separate configurations, isolated training data, or redundant quality assurance processes per language. The result is a consistent, high-quality customer experience globally, with significantly lower operational complexity.



## Enterprise security & conversational guardrails

In a live contact center, an AI needs strict rules of engagement. Generalist models are designed to be “helpful” at all costs, which leaves them vulnerable to prompt injections and policy workarounds. Raven 3.5 is different. It is built from the ground up with enterprise-grade guardrails to guarantee brand safety, data privacy, and factual accuracy.

- **Structural jailbreak resistance:** Raven 3.5 is highly resistant to prompt injection. It separates system instructions from user inputs at the token level, treating system prompts as authoritative rules and user messages as untrusted input. We actively train the model against synthetic adversarial attacks—like attempts to extract internal tool schemas—ensuring it responds with safe, polite refusals.
- **Strict out-of-domain (OOD) detection:** Raven 3.5 is trained to know the exact boundaries of its knowledge. If a caller asks an off-topic question, the model flags the turn with a special OOD token. This prevents hallucinations, triggers clean handoffs to human agents, and gives your team a data trail to identify recurring gaps in your knowledge base.
- **Data integrity & PII redaction:** We never train on public internet data; Raven learns exclusively from verified production logs. Before training, our strict redaction pipeline scrubs Personally Identifiable Information (PII) and replaces it with realistic synthetic data to protect customer privacy. Furthermore, raw model outputs are never used as training targets—every example is rigorously relabeled to ensure the model only learns from perfect responses.
- **Voice-specific behavioral penalties:** Beyond security, our training pipeline actively penalizes the AI for behaviors that ruin live voice calls. Raven is explicitly trained to avoid sycophantic openers (“I would be absolutely delighted to help you with that!”), echoing previous messages verbatim, or using ambiguous gender terms in languages like Spanish and Polish.



# As the Raven flies: The roadmap

Raven 3.5 represents PolyAI's current production deployment, proven, optimized, and powering ~90% of Agent Studio implementations. The next major architectural evolution, Raven Omni, is currently in research and development with no set release date.

## Raven today

Raven 3.5 uses a pipeline: speech → text → LLM → text → speech. This delivers production-grade performance today, but the speech-to-text conversion necessarily discards information, such as tone, emotional inflection, emphasis, background noise context. These signals contain valuable information for understanding customer intent and state.

## Raven research direction: audio-native understanding

PolyAI's research team is developing an audio-native architecture where the model processes audio features directly alongside transcribed text. An audio encoder would give Raven acoustic awareness in addition to textual understanding, hearing how something was said, not just what was said.

### Potential improvements this architecture could enable:

- Better accent handling without dialect-specific training
- Emotional awareness (detecting frustration, confusion, satisfaction)
- Improved disambiguation in noisy environments
- More natural handling of speech disfluencies and overlapping speech

This remains a text-output model (feeding TTS), but with richer input understanding. Release timing depends on research progress and production validation requirements.

## Beyond Omni: speech-to-speech

Longer-term research explores full speech-to-speech architectures—generating audio directly without TTS.

This could offer lower latency, more natural prosody, and emotionally responsive delivery. This capability remains in early research stages with no defined timeline.

## Why the roadmap matters

Investing in Raven means partnering with PolyAI's ongoing R&D program, not just licensing a static model. The architectural decisions made for voice-first design—specialized training, production-focused optimization, domain-specific data collection—compound over time. Each advancement builds on the previous foundation rather than requiring architectural restarts.

PolyAI's deployment concentration (~90% of implementations use Raven) creates a unique research advantage: production conversations at scale provide training signal and edge case discovery that pure research environments cannot replicate. As the model improves, more deployments choose Raven; more deployments generate more training data.

Critical distinction: Raven 3.5 is production-ready today with proven reliability across diverse deployments. The next update, Raven Omni, represents future research directions that leverage PolyAI's existing architectural advantages, not a requirement for current production deployments.

### Critical distinction:

Raven 3.5 is production-ready today with proven reliability across diverse deployments. The next update, Raven Omni, represents future research directions that leverage PolyAI's existing architectural advantages, not a requirement for current production deployments.



# Decision framework: Raven vs. other models

**We never lock our clients into using Raven.** But when you move from the lab to the real world, architecture matters. Here is exactly when to use a specialized customer service model, and when to rent a generalist API like ChatGPT or Claude.

## Choose Raven (The Specialist) when:

- **You are building for production**, not just testing in an R&D sandbox.
- **Latency will kill the experience.** You are building multi-turn conversations where a 2-second delay on every response compounds into dead air or abandoned chats.
- **You need real work done.** The AI must reliably execute complex backend workflows and API calls while the user is engaging.
- **You are deploying globally.** You need perfect language consistency and the same high IQ across 23 languages.
- **Your volume demands scale.** You need dedicated infrastructure that won't slow down or throttle you during peak hours.

## Choose ChatGPT or Claude (The Generalists) when:

- **You are just prototyping.** Getting a proof-of-concept built today matters more than enterprise-grade reliability.
- **You need broad, random knowledge.** Your users might ask about 16th-century history or need Python code written (rare in customer service).

## The bottom line

OpenAI and Anthropic build world-class generalists. But for enterprise customer service, renting a massive model on shared cloud infrastructure guarantees unpredictable delays. You have to write massive prompts to force them to act like customer service agents, and their generalist training often creates unnatural, robotic behaviors in a live environment.

**Raven eliminates the speed-versus-accuracy tradeoff.** By focusing entirely on customer service, it gives you the sub-300ms speed of a specialist with the domain expertise already built in.

