Domain-Adaptive Contrastive Embeddings for Frugal Skill Routing in Agentic E-Commerce Evaluation

Yue Yu[1,*], Meisam Hejazinia[1], Mark Martinov Kirichev[1] and Siamak Rajabi[1,*]

[1]*Amazon, United States*

Abstract

Large-language-model (LLM) agents increasingly operate and *evaluate* e-commerce workflows, and a recurring building block is a *skill router*: a semantic-retrieval layer that maps a free-form question to the correct skill in a catalog. The same layer lets an LLM-as-judge evaluator decide which skill *should* have answered and whether the catalog covers the question at all. In practice it runs on general-purpose embedders that mishandle the dense, abbreviation-heavy vocabulary of retail supply-chain operations, and the reflexive fixes, a larger general embedder or a frontier LLM prompted to route, are metered per call. We instead adapt a *small* open-source encoder to the domain and deliver the adaptation in the two forms a deployed system needs. The *static* recipe fully fine-tunes a 22M-parameter MiniLM-class encoder on typed contrastive pairs under a leakage-free skill-holdout split, lifting overall routing Hit@1 from 0.474 (a production 1024-d general embedder) to 0.656, and in-distribution Hit@1 from 0.393 to 0.746, over a compact 384-d index at zero marginal inference cost. The *continual-learning* recipe keeps that encoder current as the catalog grows from 23 to 43 skills: warm-starting plus a 30% replay sample holds overall Hit@1 near 0.76, matching a full retrain at roughly half the compute while avoiding catastrophic forgetting. We state the main caveat plainly: the gains concentrate on skills seen in training, and on genuinely unseen skills the general embedder still leads. A score-fusion ablation tops every arm but reintroduces a per-query general-embedder call, so we keep the standalone specialist central and point to a lightweight learned gate as future work. For the retrieval substrate of agentic e-commerce, a small specialized embedder is both more accurate in domain and far cheaper than a larger general model or a frontier LLM.

Keywords

Agentic AI for e-commerce, domain-adaptive embeddings, parameter-efficient fine-tuning, continual learning, contrastive learning, skill routing, LLM-as-judge evaluation

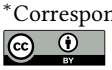
1. Introduction

Generative and agentic AI are moving from the storefront into the operational core of e-commerce. Beyond shopper-facing search and recommendation, tool-augmented LLM agents now plan multi-step workflows over enterprise operational data: they call structured-data tools, query inventory and fulfillment systems, and chain those calls into an analysis. A representative and economically important example is supply-chain *root-cause analysis* (RCA) for product availability. When an item goes out of stock, a multi-agent LLM framework reasons over operational tables, demand forecasts, and replenishment signals to explain *why*, work that otherwise consumes scarce analyst time [1]. Such systems expose their capabilities as a catalog of modular *skills*, each a small unit of analysis with a short textual specification (for example, "check whether an item is enrolled in automated demand forecasting" or "profile an item's seasonality"), and the agent answers a question by selecting and invoking the right skills. The same building block recurs across agentic e-commerce: a conversational shopping assistant routes a request to the right tool, and a customer-service agent picks its next action from a catalog of capabilities [2, 3]. Wherever an agent chooses among many skills, the quality of that choice rests on the retrieval layer we study here.

Because these agents run largely unattended, their output is itself evaluated, and a growing practice is *LLM-as-judge* evaluation [4]: a separate framework inspects an agent's trace and asks whether it invoked the appropriate skills and whether the catalog *covers* the question at all. The acting agent and its judge depend on one shared component, a *skill router* that maps a free-form question to the correct

GenAIECommerce'26: The Third Workshop on Agentic and Generative AI for E-Commerce, co-located with RecSys, September 28, 2026, Minneapolis, MN, USA

*Corresponding authors: Yue Yu (khrisyu@amazon.com), Siamak Rajabi (siamakr@amazon.com).



skill. The actor uses it to decide what to do; the judge uses it to decide what *should* have been done and to flag coverage gaps. This router is almost always nearest-neighbor search over a shared embedding space, so its quality upper-bounds both the agent's task accuracy and the reliability of the automated evaluation layered on top of it. We study this routing substrate, in the concrete setting of a production RCA evaluator, rather than the agent's reasoning policy.

The embedding model behind that search is often the limiting factor. General encoders do well on open-domain retrieval [5], but the operational language of a retail supply chain is dense with abbreviations and overloaded terms (inventory weeks-of-cover, stock-availability scores, and many program-specific acronyms) that rarely appear in their training data. A question phrased in this vocabulary is mapped near the wrong skill, and the agent confidently executes the wrong analysis. The failure is silent: a fluent, plausible answer is still returned, so it is costly and hard to detect, and the same mechanism drives both the actor's tool choice and the evaluator's coverage decision. The stakes are highest in the evaluation layer, because silent misroutes erode trust in the automated judge meant to catch agent errors, and they cannot be fixed by improving the agent's prompt, since the fault lies upstream in the retrieval geometry.

The dominant reflex is to reach for a larger model: a bigger general embedder or a frontier LLM prompted to route. Both are billed per call, and at the per-decision scale of a production evaluator that is a direct and recurring cost. This is where the problem becomes as much economic as technical: the routing layer is invoked on every question the evaluator scores, so a metered embedding or generation call is paid again and again, and, as we show, neither larger option is even the most accurate for this narrow task. We take the opposite route. We adapt a *small* open-source encoder to the domain by contrastive fine-tuning on a curated corpus, so the model stays small enough for commodity hardware, the index stays compact, inference carries no per-call charge, and the adaptation targets precisely the vocabulary the general models miss. The goal is a routing layer that is simultaneously more accurate in domain and cheaper to serve.

Realizing this frugal approach in a system that must keep running raises two problems that a one-shot benchmark hides, and they organize the paper. First, because routing and coverage are defined over the very catalog used in training, a naive evaluation can leak skill identity into the test queries and inflate the result; we therefore build a leakage-free *skill-holdout* split and report in-distribution and out-of-distribution accuracy separately (Sections 4, 5). Second, operational catalogs are not static: new skills are authored and shipped on a rolling basis, so the router must absorb them without forgetting the old ones and without paying to retrain from scratch each time; we therefore study a *continual-learning* recipe that mimics a catalog growing in weekly batches (Section 5).

Our evidence comes from a single operational e-commerce setting, supply-chain RCA and its LLM-as-judge evaluator, rather than shopper-facing search, but the retrieval layer we study is the one that lets any tool-augmented agent plan over and route among a catalog of skills. The contribution therefore speaks directly to *efficiency and scalability*, since it removes a metered per-query call from the routing path, and to *agent trust*, since it makes both the actor's tool choice and the automated evaluator's coverage verdict more reliable. The components we combine (in-batch contrastive learning, hard negatives, low-rank adaptation, experience replay) are standard; the contribution is putting them together under a leakage-aware protocol, validating the result offline inside the evaluator's real retrieval code, and showing that the same small encoder can be kept current as the catalog evolves. Concretely:

- A reproducible **dataset-construction methodology** for domain-adaptive retrieval in agentic e-commerce, producing typed contrastive pairs (terminology, skill rubric, question–skill, hard negatives, and paraphrase-invariance pairs) over a skill catalog, with a leakage-free skill-holdout split and a distribution-matched training signal drawn from real historical questions.
- A **static frugal recipe** whose central model fully fine-tunes a small (22M-parameter) MiniLM-class encoder and, on a leakage-free split, lifts overall routing Hit@1 from 0.474 (a production 1024-d general embedder) to 0.656 and in-distribution Hit@1 to 0.746, over a compact 384-d index and at zero marginal inference cost.
- A **continual-learning recipe** that keeps the same small encoder current as the catalog grows

from 23 to 43 skills: warm-start plus a 30% replay buffer holds overall Hit@1 near 0.76, matches full retraining at about half the cumulative compute, and prevents the forgetting that collapses a naive replay-free update.
- A **measured frugality analysis and ablations** covering data scale, training objective, backbone size and adaptation method (multi-seed), a lexical-router control, a coverage-verdict measurement, a score-fusion upper bound, and a head-to-head against a single-prompt frontier LLM, plus an integration proof-of-concept inside the real retrieval code of a production LLM-as-judge evaluator, all showing that a $0-marginal-cost specialized encoder (a measured 8 ms median CPU decision) is the most cost-effective option for this substrate.

We foreground one caveat from the start: the gains are concentrated on skills the model has seen, and on a strict unseen-skill arm the general embedder still leads. The method therefore helps most when the served catalog overlaps the training catalog, which is the common production regime and the one the continual-learning recipe is designed to maintain.

2. Related Work

Agentic e-commerce and LLM-as-judge evaluation. Tool-augmented and multi-agent LLM systems are applied across e-commerce, from conversational recommendation to autonomous operational analysis [2, 1, 3, 6], and strong LLMs are now routinely used to evaluate model and agent outputs [4]. These systems decompose work across skills or sub-agents and route requests among them, and production evaluators additionally use embedding retrieval to decide *coverage*, so routing quality upper-bounds end-to-end quality and evaluation reliability. We target that routing substrate rather than the agent policy.

Contrastive embeddings, PEFT, and domain adaptation. Dense sentence encoders trained with contrastive objectives are the de facto standard for semantic retrieval [7, 8], where in-batch negatives [9] and hard negatives sharpen the separation between relevant and irrelevant pairs, and compact open models [10, 11] offer strong initialization at low serving cost [5]. LoRA [12] and QLoRA [13] adapt models by training a small set of low-rank parameters. Unsupervised domain adaptation of retrievers is an active area [14, 15]; when domain gold signals already exist in historical logs, the supervised counterpart is the more label-efficient choice, so we adopt a small *supervised* corpus of typed pairs and adapt a retrieval encoder to it. Two gaps in this prior work motivate our framing. Retrieval benchmarks routinely let entities seen in training reappear at test time, which inflates apparent gains when the target labels are the very skills being adapted; we close this with a skill-holdout split and separate in/out-of-distribution reporting. And domain-adaptation studies typically stop at an offline metric, leaving open whether the gain survives a real serving stack; we instead run the adapted encoder through the actual retrieval code of a production LLM-as-judge evaluator in an offline integration test.

Continual learning and rehearsal. Neural models overwrite earlier knowledge when trained on new data, the catastrophic-forgetting problem [16]. Two families address it: regularization of important weights, such as elastic weight consolidation [17], and *rehearsal*, which replays a sample of past data while learning new tasks [18, 19, 20]. Our catalog-growth setting is a class-incremental problem in disguise, where each new batch of skills is a set of new routing targets. We use the simplest effective remedy, a small experience-replay buffer combined with warm-starting, and show that a modest replay fraction recovers almost all of the accuracy of a full retrain at a fraction of the cost, which is the property a production update cadence needs. Applying rehearsal to a domain-adapted *retrieval encoder* under a growing skill catalog, rather than to a classifier, is the specific setting we contribute.

3. Problem Setting

Skill catalog and encoder. We consider the retrieval layer of an agentic system that exposes a catalog of *skills*, capability modules each described by a short textual specification (name, description, when-to-use guidance, and rubric). Let $\mathcal{S} = \{s_1, \ldots, s_N\}$ denote the catalog, with each $s_i \in \mathcal{T}$ drawn from the space of text $\mathcal{T}$. A query $q \in \mathcal{T}$ expresses an information need that may or may not be served by the catalog. The sole learned component is an encoder $f_\theta : \mathcal{T} \to \mathbb{R}^d$ with parameters θ, used, for any text $x \in \mathcal{T}$, through its ℓ_2-normalized form

$$\hat{f}_\theta(x) \;=\; \frac{f_\theta(x)}{\|f_\theta(x)\|_2} \;\in\; \mathbb{S}^{d-1},$$

so that similarity is the cosine score $\mathrm{sim}_\theta(x, x') = \langle \hat{f}_\theta(x), \hat{f}_\theta(x') \rangle$. We write $\mathbf{u} = \hat{f}_\theta(q)$ for the query embedding and $\mathbf{v}_i = \hat{f}_\theta(s_i)$ for the skill embeddings, collected as the index $V_\theta = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$.

Downstream decisions. Two decisions consume the similarity scores. *Routing* selects the skill whose specification is most aligned with the query,

$$r_\theta(q) \;=\; \arg\max_{i \in [N]} \; \mathrm{sim}_\theta(q, s_i),$$

evaluated by whether the gold skill appears among the top-k candidates, $\mathrm{Hit@}k \;=\; \Pr_{(q,i^\star)}[\, i^\star \in \mathrm{Top}_k(q)\,]$, where $\mathrm{Top}_k(q)$ ranks skills by $\mathrm{sim}_\theta(q, \cdot)$ and $i^\star$ is the annotated skill. *Coverage* asks whether *any* skill is relevant, summarized by the margin $m_\theta(q) = \max_i \mathrm{sim}_\theta(q, s_i)$, which a fixed thresholding rule maps to an ordinal verdict $g : [-1, 1] \to \{\textsc{full}, \textsc{partial}, \textsc{weak}, \textsc{none}\}$ via cutoffs $\tau_{\textsc{full}} > \tau_{\textsc{part}} > \tau_{\textsc{weak}}$. Coverage quality is measured by agreement with human labels.

The intervention and its attribution. The host system is a fixed map Φ that consumes embeddings to produce routing and coverage outputs; the deployed pipeline is the composition $\Pi_\theta = \Phi \circ \hat{f}_\theta$. Index construction, the scoring function, the thresholds, and the catalog are all held fixed, so Φ does not depend on θ. Replacing a baseline encoder f_{θ_0} with an adapted encoder f_θ changes Π only through the embeddings, so any change in a downstream metric $\mathcal{M}$, $\Delta\mathcal{M} = \mathcal{M}(\Phi \circ \hat{f}_\theta) - \mathcal{M}(\Phi \circ \hat{f}_{\theta_0})$, is attributable to the encoder alone. Operationally, the substitution occurs at a single configuration seam (the encoder identifier in the host config), which makes the intervention both controlled and deployable without modifying the surrounding system.

Adaptation target. We adapt f_{θ_0} into f_θ by fine-tuning on a corpus of *typed* pairs $\mathcal{D} = \{(a_j, b_j, t_j, y_j)\}_{j=1}^{M}$, where $a_j, b_j \in \mathcal{T}$, t_j indexes the pair type (question–skill, paraphrase, term–definition, sibling disambiguation, and so on), and $y_j \in \{+, -\}$ marks a positive or hard-negative relation. Anchors a_j stand where the query q stands at serving time (most are historical questions) and the paired texts b_j stand where the skill specifications s_i stand, so the geometry shaped in training is exactly the one r_θ and m_θ consume. By default we update all parameters of the small encoder (full fine-tuning); when the trainable-parameter budget is tight we instead learn a low-rank increment $\theta = \theta_0 \oplus \Delta$ (LoRA) with $|\Delta| \ll |\theta_0|$. For an anchor a with positive b^+ and negative set $\mathcal{N}(a)$, we minimize a temperature-scaled contrastive (InfoNCE) objective

$$\mathcal{L}(\theta) = -\sum_{(a,b^+)} \log \frac{\exp(\mathrm{sim}_\theta(a, b^+)/\tau)}{\exp(\mathrm{sim}_\theta(a, b^+)/\tau) + \sum_{b^- \in \mathcal{N}(a)} \exp(\mathrm{sim}_\theta(a, b^-)/\tau)},$$

with temperature $\tau > 0$ and $\mathcal{N}(a)$ combining in-batch and mined hard negatives. This pulls domain-equivalent texts together and pushes near-miss siblings apart on $\mathbb{S}^{d-1}$, shaping the geometry that r_θ and m_θ rely on.

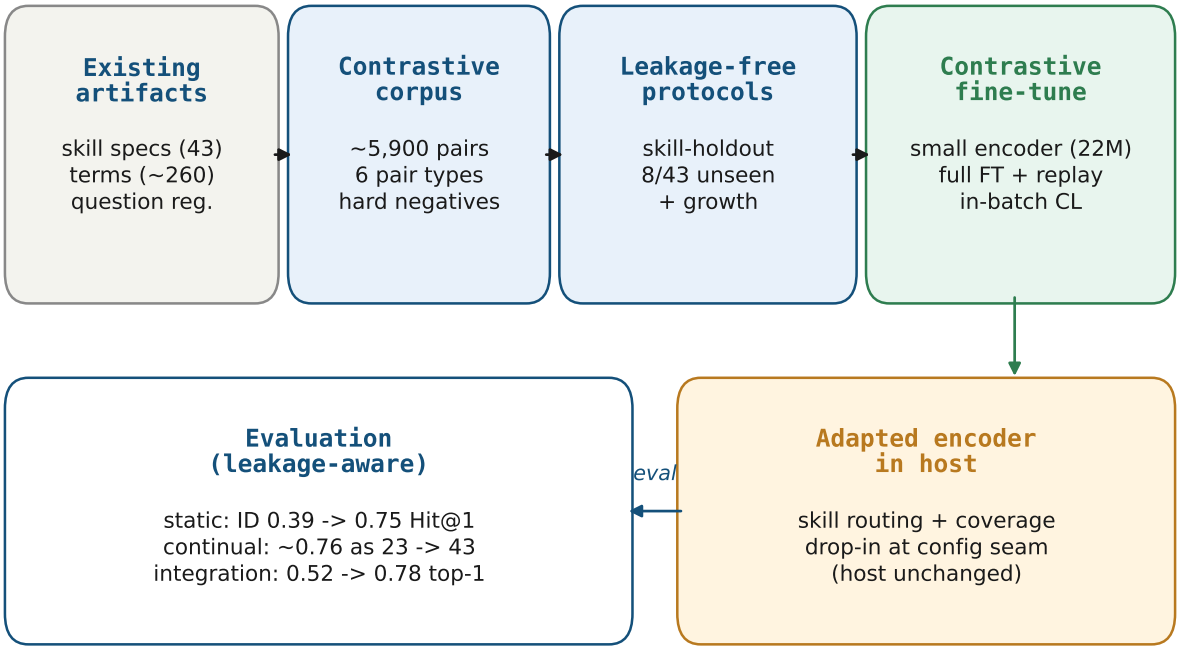


Figure 1: End-to-end pipeline. Existing skill specifications, a curated term inventory, and a historical question registry are compiled into a typed contrastive corpus and used to adapt a small encoder with contrastive fine-tuning. The same corpus supports two evaluation protocols: a leakage-free skill-holdout split for the static recipe, and a catalog-growth simulation for the continual-learning recipe. The adapted encoder is scored on frozen benchmarks and dropped into the host evaluator's real routing code with no host-side changes.

Leakage-aware evaluation. Because routing and coverage are defined over the same catalog used in training, naive splits can leak skill identity from $\mathcal{D}$ into the test queries. We evaluate under a skill-holdout protocol: let $\mathcal{S}_{\text{tr}}$ and $\mathcal{S}_{\text{te}}$ partition the catalog with $\mathcal{S}_{\text{tr}} \cap \mathcal{S}_{\text{te}} = \varnothing$, require every training pair to be anchored in $\mathcal{S}_{\text{tr}}$, and bar held-out skills even as hard negatives, so no unseen-skill text leaks through a negative. In-distribution queries target $\mathcal{S}_{\text{tr}}$ (seen skill, unseen phrasing) and out-of-distribution queries target $\mathcal{S}_{\text{te}}$ (never-seen skill).

The catalog-growth setting. Operational catalogs grow over time. We model this as a sequence of rounds $k = 1, \ldots, K$ in which the catalog expands $\mathcal{S}^{(1)} \subset \mathcal{S}^{(2)} \subset \cdots \subset \mathcal{S}^{(K)}$ by a batch of new skills each round. At round k we may update the encoder from the previous checkpoint θ_{k-1} using training pairs for the new skills together with a replay sample of pairs from the prior catalog, producing θ_k. Two quantities matter at each round: routing accuracy over the *current* catalog $\mathcal{S}^{(k)}$, and *retention*, the accuracy on the skills introduced in round 1 as later rounds accrue, which measures forgetting. A brand-new skill has no accumulated real questions in the round it debuts, exactly the cold-start the router faces in production. This setting is the dynamic counterpart of the static split above: the skills that debut in round k occupy the position of $\mathcal{S}_{\text{te}}$ at the moment they arrive (a specification in the catalog, no real-question history), and the continual recipe is the mechanism that moves them into $\mathcal{S}_{\text{tr}}$ as their pairs enter training.

4. Dataset Construction

We looked for a public corpus before building one: open retrieval suites such as MS MARCO and the BEIR/MTEB collections [5] target open-domain or shopper-facing search, and public e-commerce query datasets pair shopper queries with products, not analyst questions with operational skills; none covers vocabulary like weeks-of-cover, in-stock health scoring, or program-specific procurement acronyms.

So we construct the corpus ourselves (Figure 1). The pipeline reads from two artifact types that already exist in an agentic system, machine-readable skill specifications and a registry of historical questions, and emits typed contrastive pairs over a combined catalog of 43 operational skills. Several skills across the source packages are *semantic siblings*, near-duplicates that share intent but differ in scope; we retain them deliberately, as they are the hardest negatives in the domain and a realistic source of routing confusion.

Why typed supervised pairs. We considered three ways to obtain in-domain training signal. (i) *Unsupervised domain adaptation*, for example generative pseudo-labeling [14] or reranker distillation [15], needs no labels but yields noisy positives and, decisively, cannot use the gold question–skill routes the registry already records. (ii) *Fully synthetic* LLM-generated pairs scale cheaply but tend to paraphrase the source text, reinforcing surface vocabulary rather than the hard sibling distinctions that drive routing errors, and they risk importing the same general-model biases we are trying to correct. (iii) *Curated typed pairs* from existing artifacts, the option we adopt, make the most label-efficient use of assets an agentic system already maintains and let us deliberately over-represent the hard cases. The trade-off is scale, hand-curated supervision is smaller than a synthetic dump, which is exactly why we measure a data-scale curve (Section 5) rather than assume sufficiency.

Distribution-matched signal. An early version of the corpus drew its anchors from skill specifications ("what is X") and, although it beat the general embedder in domain, it did not generalize to the phrasing of real analyst questions, which are predominantly "why did X happen" narratives. The fix that mattered was not more skill coverage but *matching the training distribution to the question distribution*: we draw training anchors from real historical questions wherever they exist, and reserve a disjoint set of real questions for evaluation. This single change is what lifts the real-question arm, and we treat real questions as the ground-truth signal throughout.

Terminology inventory and typed pairs. We mine a curated inventory of roughly 260 in-domain terms and abbreviations from the skill specifications and supporting documentation, each paired with its gloss and equivalent surface forms (an acronym, its expansion, and a colloquial phrasing), yielding paraphrase-invariance and term–definition pairs. The full corpus contains on the order of 5,900 typed pairs across six types, designed so the objective sees both fine-grained vocabulary signal and realistic routing decisions: **term–definition** (a term and its gloss), **rubric–skill** (a skill's rubric and its specification), **question–skill** (a historical question and the skill that answers it), **paraphrase-invariance** (two surface forms of one concept), **sibling-collision** (cross-package near-duplicate skills, used as hard negatives), and **wrong-skill negatives** (a question paired with a plausible-but-incorrect skill). Hard negatives (sibling-collision and wrong-skill) make up the majority of pairs (58%), which we found essential for separating near-identical skills.

Representative pairs. Table 1 grounds the six types in one sanitized example each. The objective pulls each anchor a toward its positive b^+ and away from the hard negative b^-. The two negative-defining types carry most of the routing signal: a *sibling-collision* separates near-duplicate skills that differ only in scope (a live inventory snapshot versus a historical trend), and a *wrong-skill negative* pairs a question with a plausible but incorrect skill, so adaptation sharpens precisely the boundaries that produce routing errors.

Two evaluation protocols. The same corpus supports the two recipes. For the **static** recipe we apply the skill-holdout protocol of Section 3: 8 of the 43 skills are removed from training entirely, not as a positive and not even as a hard negative, so only their specification sits in the catalog, exactly like a newly-added skill, and their questions form a genuinely never-seen out-of-distribution arm. Training uses 4,743 pairs; the test set is 308 questions, 201 in-distribution (held-out questions of trained skills) and 107 out-of-distribution (questions of the 8 held-out skills), with train/test overlap verified to be zero.

Table 1
One representative pair per type from the contrastive corpus (Section 4). Skill specifications are referenced by name (`skill-name`) and glosses are abridged; positives b^+ are pulled toward the anchor a and hard negatives b^- are pushed apart. Examples are illustrative sanitized analogs: skill names, terms, and numeric values are generic stand-ins, with no internal identifiers, table names, or figures. The term and paraphrase types rely on in-batch negatives, marked "n/a".

Type	Anchor $a \rightarrow$ positive b^+	Hard negative b^-
Term–definition	*OOS-rate* → "share of days an item is orderable yet unavailable"	"out of stock" (generic flag)
Rubric–skill	"does the answer cover model-vs-actual demand accuracy and allocation?" → `analyze-forecast-accuracy`	`check-replenishment-enrollment`
Question–skill	"What is the trailing-12-month contribution margin and unit volume?" → `analyze-unit-economics`	`check-profitability-flag`
Paraphrase–invariance	"product group, size tier, procurement model?" → "product family, size band, purchasing model"	n/a
Sibling-collision	"current cover at each site right now?" → `summarize-stock-cover` (live)	`track-stock-cover-history` (historical)
Wrong-skill negative	"responsible buyer or team?" → `identify-item-owner`	`recommend-replenishment-action`

For the **continual-learning** recipe we instead grow the catalog in five rounds ($23 \rightarrow 28 \rightarrow 33 \rightarrow 38 \rightarrow 43$ skills) and, at each round, evaluate over the current catalog on a query-level held-out mix of real and synthetic questions; newly-added skills are synthetic-only in their debut round, matching the production cold-start. Real analyst questions are the ground-truth arm; synthetic questions supplement coverage for new and real-sparse skills and are held out from training to avoid circularity.

5. Experiments

5.1. Setup

Model and baselines. Our central model is a 22M-parameter, 384-dimensional MiniLM-class sentence encoder that we fully fine-tune on the contrastive corpus; this is the configuration we recommend and report throughout. Cheaper or larger variants enter only as ablations (Section 5.4): low-rank adaptation (LoRA) of the same encoder, and a larger 110M-parameter general-purpose backbone. The baselines are the base encoder, a production general-purpose multimodal embedder (1024-d) that represents the deployed status quo, a second open 110M encoder used zero-shot, and a frontier closed-weight instruction-tuned LLM prompted to route.

Objective and training. We train with the in-batch contrastive (multiple-negatives ranking) objective [9]: each anchor's paired positive is pulled together while all other in-batch documents (and the curated hard negative, when present) are pushed apart, under a temperature-scaled cross-entropy over cosine similarities. All arms share an identical configuration: the AdamW optimizer, a maximum sequence length of 256 tokens, and InfoNCE temperature $\tau{=}0.05$. The full-fine-tune arms train for 3 epochs at learning rate 2×10^{-5}; the LoRA arms train for 5 epochs at 5×10^{-4} with rank $r \in \{8, 16\}$, $\alpha{=}2r$, and dropout 0.05, injected into the query/key/value projections. Training is short: the corpus fits in memory and a complete fine-tune finishes in minutes on a single commodity accelerator and runs to completion on CPU, consistent with the frugality goal and with routine re-adaptation. At inference the model is a standard sentence encoder and the index is rebuilt by embedding each skill specification once.

Table 2
Static recipe on the leakage-free skill-holdout benchmark (n=308: 201 in-distribution, 107 out-of-distribution; 8 of 43 skills held out entirely). Hit@1 overall and by arm; brackets are 95% Wilson intervals (bootstrap intervals agree to three decimals where per-question predictions are retained). The lexical row is a TF–IDF router over the same specifications, testing whether raw domain signal suffices without learning. The specialist is our 22M full fine-tune; the fusion row is an ablation (Section 5.4) that also calls the general embedder per query and reports the best blend weight from a sweep (α=0.6), so it is an upper bound. Best standalone value per column in bold.

Model	dim	overall	in-dist.	out-of-dist.
Lexical TF–IDF router	sparse	0.283 [.24, .34]	0.234 [.18, .30]	0.374 [.29, .47]
Base encoder (zero-shot)	384	0.328 [.28, .38]	0.293 [.23, .36]	0.393 [.31, .49]
Open 110M encoder (zero-shot)	768	0.357 [.31, .41]	0.303 [.24, .37]	0.458 [.37, .55]
General embedder (production)	1024	0.474 [.42, .53]	0.393 [.33, .46]	**0.626** [.53, .71]
Specialist (ours, 22M full-FT)	384	**0.656** [.60, .71]	**0.746** [.68, .80]	0.486 [.39, .58]
Fusion (specialist + general), ablation	n/a	*0.731* [.68, .78]	*0.751* [.69, .81]	*0.692* [.60, .77]

Why full fine-tuning, and why LoRA as the frugal alternative. At this scale full fine-tuning is affordable, so it is our default. We include LoRA because it adds no serving-time latency (the low-rank update merges into the base weights) and is architecture-agnostic, unlike adapter or prefix-tuning families that add inference cost. We do not use QLoRA [13]: its 4-bit quantization exists to fit multi-billion-parameter models into limited memory, a constraint absent at the $\leq$110M scale here, where it would only add quantization error for no memory benefit.

Evaluation design. Following the principle that a routing layer should be judged on the decisions the host actually makes, we combine several kinds of evidence rather than a single number. For the static recipe we report routing Hit@1 split into in-distribution and out-of-distribution arms, plus a data-scale curve. For the continual-learning recipe we report per-round Hit@1 over the growing catalog, a retention (forgetting) curve on the original skills, a replay-fraction ablation, a cost accounting, and a multi-seed robustness check. Across both we separate the real-question arm (the ground-truth production signal) from synthetic coverage questions, and we complement the offline numbers with an in-situ integration test inside the host's real retrieval code. Gold routes come from an LLM annotator whose agreement with a set of 96 human labels is micro-F1 0.63 (exact skill match) and 0.69 after normalizing a few unambiguous skill renames; we therefore read *absolute* numbers as carrying label noise of about that size and lean on *relative* comparisons scored against the identical gold. Every reported proportion carries a 95% Wilson interval; where per-question predictions are retained, bootstrap resampling (10,000 draws) reproduces the Wilson brackets to three decimals, and we add significance tests for the headline comparisons (Section 5.2). All embedding arms use identical index and scoring code.

5.2. Static recipe: leakage-free routing and the in/out-of-distribution split

Table 2 reports routing Hit@1 on the 308-question skill-holdout benchmark. Out of the box no baseline clears 0.48 overall: the base encoder sits at 0.328, an open 110M encoder at 0.357, and the production general embedder at 0.474. Fully fine-tuning the small encoder is what closes the gap, reaching 0.656 overall, a +0.182 lift over the general embedder at roughly one-third of its vector width and no per-call cost. The gain is concentrated in distribution: on the 201 in-distribution questions the specialist reaches 0.746, nearly double the general embedder's 0.393. Both headline gaps are statistically robust: specialist versus general embedder gives z=4.55, p=5$\times10^{-6}$ overall and z=7.15, $p<10^{-12}$ in distribution (two-proportion tests), and the intervals in Table 2 are far from overlapping. The lexical control sharpens the attribution: a TF–IDF router over the very same specifications manages only 0.283 overall, below even the zero-shot base encoder, so the gain comes from *learned* domain adaptation, not from merely having any in-domain signal.

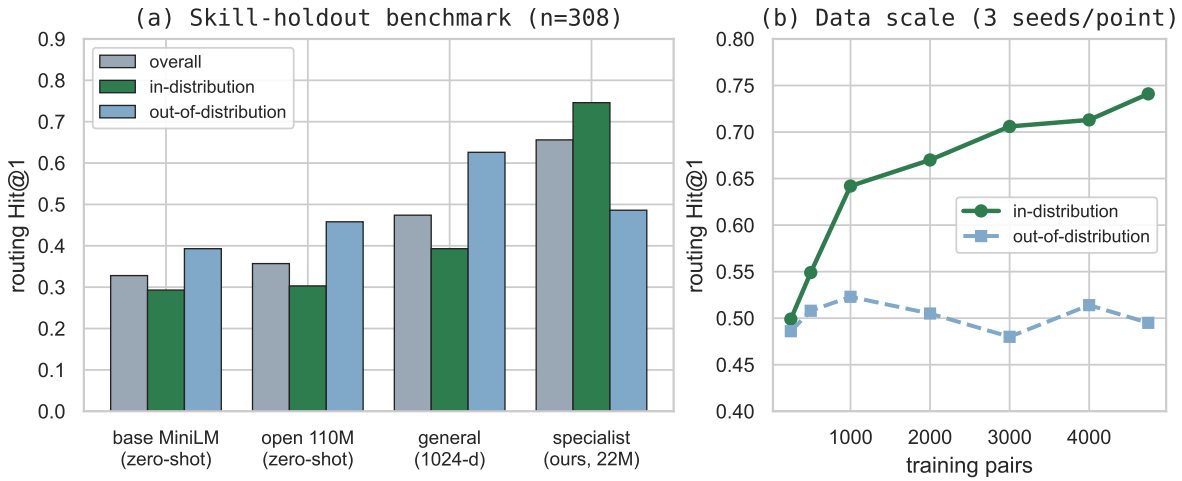


Figure 2: Static recipe. (a) Routing Hit@1 by arm on the skill-holdout benchmark: the specialist leads overall and in-distribution, while the general embedder leads out-of-distribution. (b) Data-scale curve (three seeds per point): the in-distribution arm climbs and is still rising at the full corpus, while the out-of-distribution arm is flat near 0.50.

The honest out-of-distribution finding. On the 107 genuinely unseen questions the ordering flips: the general embedder leads at 0.626 and the specialist trails at 0.486. This is the central caveat of the static recipe, and two observations place it correctly. First, it is not catastrophic forgetting: the specialist actually *improves* out-of-distribution over its own zero-shot base ($0.393 \rightarrow 0.486$), so fine-tuning helps even on unseen skills; the general embedder's larger 1024-d space is simply stronger on skills no small model has seen, and no small open base we tried closes that gap. Second, it is a ranking bias toward trained skills rather than a loss of unseen representation: fine-tuning sharpens seen skills so strongly that, in a strict argmax over the whole catalog, a seen skill often outranks the correct unseen one. The deficit is itself statistically significant (z=2.06, p=0.039), so we report it as a real property of the method, not benchmark noise.

What this means in practice. The static specialist is the right tool when the served catalog overlaps the training catalog. That is the common production regime, and keeping it true over time is precisely the job of the continual-learning recipe (Section 5.3); a brand-new skill, until its pairs enter training, is better served by the general embedder or the fusion ablation.

Data scale. Figure 2 traces routing Hit@1 as the curated corpus grows, at three seeds per point on the fixed skill-holdout split. The in-distribution arm climbs steadily, from 0.499 at 250 pairs to 0.741 at the full 4,743, and is still rising at the top, so the corpus is adequate for the reported gains but demonstrably not saturated, and additional curated pairs remain the largest available lever. The out-of-distribution arm stays flat near 0.50 across the entire nineteen-fold data range. The two curves together are the specialization-versus-robustness trade-off made visible: more in-domain data buys in-distribution accuracy and does nothing for genuinely unseen skills, which is consistent with the ranking-bias reading above.

5.3. Continual-learning recipe: keeping the router current as the catalog grows

A static checkpoint answers the wrong question for a live system, because the catalog does not hold still. We simulate a catalog growing in five weekly rounds from 23 to 43 skills and compare four update strategies against the two un-adapted baselines, scoring routing Hit@1 over the current catalog at each round (Table 3). The strategies are: **frozen**, fine-tune once at round 1 and never update; **no-replay**, warm-start each round and fine-tune only on the new skills; **full retrain**, retrain from the base encoder

Table 3
Continual-learning recipe: routing Hit@1 as the catalog grows from 23 to 43 skills. Top block is overall Hit@1 (real plus synthetic questions, n grows $264 \to 379$); bottom block is the real-question arm (n grows $173 \to 208$), the ground-truth production signal. The fusion row is an ablation (Section 5.4) that also calls the general embedder per query; it reports the per-round best blend weight from an α sweep ($\alpha \in [0.5, 0.7]$), so it is an upper bound. Best standalone value per column in bold. Week-5 95% Wilson intervals: specialist 0.757 $[0.712, 0.798]$ overall, 0.663 $[0.597, 0.724]$ real; full retrain 0.768 $[0.723, 0.808]$ / 0.697 $[0.632, 0.756]$; frozen 0.707 $[0.659, 0.751]$ / 0.654 $[0.587, 0.715]$; no-replay 0.710 $[0.662, 0.753]$ / 0.611 $[0.543, 0.674]$.

Model	wk1 (23)	wk2 (28)	wk3 (33)	wk4 (38)	wk5 (43)
Overall Hit@1					
base encoder (zero-shot)	0.417	0.465	0.465	0.459	0.464
general embedder	0.625	0.625	0.620	0.611	0.552
frozen (train wk1 only)	0.792	0.761	0.757	0.718	0.707
no-replay	0.792	0.741	0.726	0.693	0.710
full retrain	0.792	0.758	**0.796**	0.761	**0.768**
specialist (warm+30% replay)	0.792	**0.774**	0.778	**0.775**	0.757
fusion (ablation)	*0.814*	*0.811*	*0.815*	*0.811*	*0.807*
Real-question Hit@1					
base encoder (zero-shot)	0.249	0.300	0.293	0.294	0.289
general embedder	0.497	0.495	0.480	0.471	0.389
frozen (train wk1 only)	0.717	0.690	0.692	0.662	0.654
no-replay	0.717	0.642	0.631	0.608	0.611
full retrain	0.717	0.668	**0.717**	0.681	**0.697**
specialist (warm+30% replay)	0.717	**0.695**	0.702	**0.701**	0.664
fusion (ablation)	*0.734*	*0.742*	*0.742*	*0.745*	*0.721*

on the entire current catalog each round; and the **specialist**, our recommended recipe, which warm-starts from the previous checkpoint and fine-tunes on the new skills plus a 30% replay sample of the prior catalog.

Every update strategy degrades, but not equally. As the catalog grows there are more candidates to confuse, so every model drifts down. The un-adapted baselines start low and the general embedder falls furthest by week 5 (overall 0.552, real 0.389), because the newest skills are exactly the ones its general vocabulary handles worst. Among the update strategies, the specialist holds overall Hit@1 at or above 0.757 every week and full retrain is its close upper bound (0.768 at week 5), while the two weak strategies reveal mirror-image failure modes. The frozen model, whose weights never change, cannot learn the new skills (its week-5 new-skill arm is 0.58) but preserves the old ones, landing at 0.707. No-replay learns the new skills well but forgets the old ones, landing at 0.710: essentially tied with freezing, and on the production-critical real arm actually worse (0.611 versus 0.654), because the real arm is dominated by older skills that no-replay overwrites. The specialist avoids both traps and its real-arm accuracy (0.664 at week 5) trails only the full retrain (0.697).

We read the accuracy rows of Table 3 conservatively. At week 5 the specialist's real-arm lead over the frozen model is two questions (0.663 $[0.597, 0.724]$ versus 0.654 $[0.587, 0.715]$), well inside the intervals, so the per-week accuracy rows alone do not separate the update strategies at this sample size. What separates them is *retention* (Figure 3b): on the original 23 skills at week 5, no-replay falls to 0.602 $[0.542, 0.659]$ while 30% replay holds 0.727 $[0.671, 0.777]$, non-overlapping intervals, with the full retrain at 0.739. The defensible claim is therefore not that replay beats freezing by a point of week-5 accuracy; it is that replay is what lets a weekly update learn new skills *without* the forgetting that a replay-free update provably suffers.

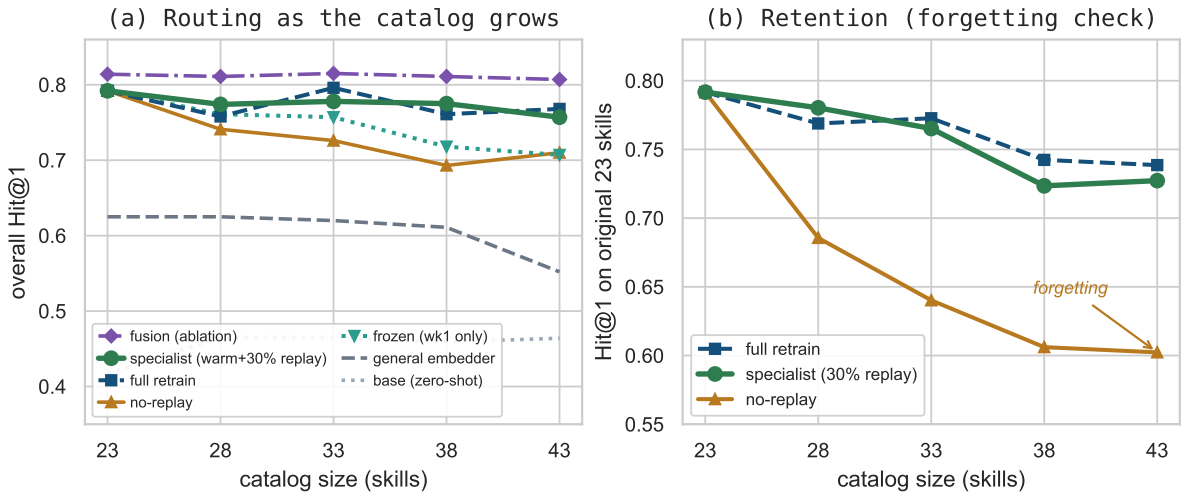


Figure 3: Continual-learning recipe. (a) Overall Hit@1 as the catalog grows from 23 to 43 skills: the specialist tracks the full-retrain upper bound, while the frozen and no-replay updates converge to a lower plateau and the general embedder declines. (b) Retention on the 23 original skills: no-replay forgets (down to 0.60), while 30% replay holds near the full-retrain line.

Replay prevents forgetting, and 30% is the sweet spot. Figure 3 shows retention on the 23 original skills as the catalog grows. No-replay collapses them from 0.79 to 0.60, one week's update overwriting the last; a 30% replay buffer holds them at 0.73, tracking the full-retrain upper bound of 0.74. Sweeping the replay fraction over $\{0, 10, 20, 30, 50, 100\}\%$ at the full catalog gives overall Hit@1 of 0.710/0.749/0.731/0.757/0.744/0.768: 30% is the best of the incremental options and the only fraction that stays at or above 0.75 every week, while more replay mostly adds cost (100% is a full retrain). Two honesty notes on this sweep: it is non-monotonic, and the 30%-versus-10% gap (0.757 versus 0.749, three questions of 379) sits inside the three-seed band of the recipe itself (0.744–0.757). The decisive contrast is therefore replay-versus-none, which the retention curve settles; the specific fraction is a weakly-identified operating point, and any modest value in the 10–50% range is defensible. This matches the continual-learning literature, where modest rehearsal buffers are typically sufficient [18, 20].

Cost and robustness. The point of the specialist is that it buys the retrain's accuracy without the retrain's cost. Cumulative training time to keep the router current across all five weekly updates is about 839s for warm-start plus 30% replay versus about 1712s for a full retrain each week, roughly a $2\times$ saving. We state the absolute size plainly: at today's 43-skill catalog this is minutes per week, so the saving is an argument about scaling behavior rather than present-day spend; full retraining grows with the entire catalog every week while the incremental update grows only with the new skills and a small replay sample, so the ratio widens as the catalog and pair corpus grow. Repeating the specialist recipe across three seeds preserves the model ordering and keeps the full-catalog overall Hit@1 in a narrow band, 0.744–0.757 (mean 0.752), so the recipe is stable rather than seed-sensitive. A periodic full retrain (for example monthly) remains worthwhile to reclaim the last point of retention, but is too expensive for the weekly cadence.

5.4. Ablations, fusion, and frugality

Fusion is an accuracy upper bound with a per-query cost. The static out-of-distribution gap and the general embedder's day-one coverage of brand-new skills both suggest blending the two encoders. A score-fusion router, $\alpha\, z(\text{specialist}) + (1-\alpha)\, z(\text{general})$ with per-query z-scoring and argmax over the catalog, tops every arm. On the static skill-holdout benchmark it reaches 0.731 overall with in-distribution 0.751 and out-of-distribution 0.692, beating both parts on both arms (Table 2); in

Table 4
Recipe ablations on the 89-question development benchmark (75 seen / 14 unseen; earlier catalog snapshot). Overall Hit@1, mean ± sample standard deviation over five seeds. All arms train on the same typed pairs with the same protocol as Section 3. The objective row uses the same five seeds, backbone, and full fine-tune as the specialist row, changing only the training loss, so the two rows differ in nothing but the objective. Best arm in bold.

Arm	Backbone / method	overall Hit@1
Specialist recipe	MiniLM-class 22M, full FT	$\mathbf{0.742 \pm 0.008}$
Top-2-layers FT	MiniLM-class 22M	0.688 ± 0.022
LoRA r=16	MiniLM-class 22M	0.649 ± 0.026
LoRA r=8	MiniLM-class 22M	0.640 ± 0.026
Larger backbone, full FT	open 110M encoder	0.674 ± 0.026
Larger backbone, LoRA r=16	open 110M encoder	0.685 ± 0.052
Second family, full FT	open 33M encoder	0.658 ± 0.017
Objective ablation	22M full FT, cosine loss	0.548 ± 0.020

the catalog-growth setting it holds overall Hit@1 near 0.81 and real-question Hit@1 near 0.73 every week, above every standalone model (Table 3). We nonetheless keep it out of the central contribution, because fusion calls the general embedder on *every* query and so gives back the frugality that motivates the whole approach. The right way to capture most of its benefit cheaply is a lightweight learned gate that spends a general-embedder call only on the queries that need it, which we discuss in Section 6.

Adaptation method, objective, and backbone. Table 4 reports the recipe ablations. They were run on the 89-question development benchmark that guided recipe design (same corpus construction and held-out-skill arm, an earlier catalog snapshot; 75 seen / 14 unseen questions), so we use them for *relative* comparisons and keep the leakage-hardened 308-question protocol for the headline numbers. Three findings. First, the backbone comparison is now fine-tune versus fine-tune, and bigger still does not win: the 110M encoder *fully fine-tuned on the same pairs* reaches 0.674 ± 0.026, below the 22M specialist's 0.742 ± 0.008, and a 33M sibling from a second open family lands between (0.658 ± 0.017). Zero-shot, the same 110M encoder manages only 0.357 on the main benchmark (Table 2), so neither scale alone nor scale plus adaptation beats the small specialist here. Second, adaptation method: LoRA at rank 8 or 16 gives up 0.09–0.10 of overall Hit@1 against full fine-tuning at this model scale, and fine-tuning only the top two layers recovers part of that gap (0.688 ± 0.022); at 22M parameters, full fine-tuning is both affordable and the accuracy choice, which is why LoRA is the frugal fallback rather than the default. Third, the objective: this ablation is anchored on the best model, holding the specialist's backbone, full fine-tune recipe, data, and all five seeds fixed and changing only the loss. Replacing the in-batch ranking objective with a cosine-embedding regression loss costs 0.19 overall Hit@1 ($0.742 \pm 0.008 \rightarrow 0.548 \pm 0.020$, five seeds each; the gap is nearly ten times the larger seed band): regressing each pair's similarity in isolation never teaches the encoder to rank the correct skill *above* its near-siblings, which is precisely the argmax competition routing runs, so a ranking loss is the right match for the decision. The specialist recipe is also the most seed-stable arm in the table (sample standard deviation 0.008 over five seeds, versus 0.017–0.052 for the others).

Coverage verdicts, measured. Section 3 formalizes the coverage decision through the margin m_θ and a thresholded verdict map; we report its measurement here. On a 76-item coverage benchmark with human verdict labels (thresholds calibrated per encoder on a disjoint 51-item set), an adapted encoder from our recipe reaches accuracy 0.684 with Cohen's κ=0.42; the single-prompt frontier LLM reaches 0.645 with κ=0.50; and even a calibrated lexical TF–IDF margin reaches 0.684 with κ=0.41. The reading is that coverage is *threshold-dominated*: after per-model calibration, the choice of encoder moves verdicts far less than it moves routing, which is why routing Hit@1 is our discriminating

metric and why we treat coverage thresholds as deployment configuration to recalibrate per encoder (Section 6).

Frontier LLM and the frugality frontier. We benchmark a frontier instruction-tuned LLM as a direct router over the production catalog using a single fixed prompt. It reaches routing Hit@1 0.528, within noise of the general embedder and below the specialist. Because this is a single-prompt baseline, we read it as a lower bound on what prompt engineering could reach and restrict the comparison to routing Hit@1, but the economic point is prompt-independent and is the main one. The three options sit at distinct per-decision cost tiers, and we anchor each tier with a measurement or a public list price. The specialist routes a query in a measured 8.1 ms median (p_{90} 8.9 ms) on an 8-thread commodity CPU at batch size one, including the argmax over the 43-skill index, after a one-time 0.34 s index build; its marginal cost is zero. The general embedder is a metered embedding-API call on every query; representative public list prices for embedding APIs are \$0.02–\$0.13 per million tokens[1], tiny per call but metered, rate-limited, and paid on every routed question at fleet scale. The frontier LLM pays per token on a prompt that must carry the catalog: our fixed router prompt is $\sim$2.8k input tokens (the 43-skill catalog text plus the question and instructions) with $\sim$25 output tokens, about \$0.014 per routing decision at current public frontier list prices (\$5 input / \$25 output per million tokens)[2], four to five orders of magnitude above one embedding call. The cost we deliberately do not meter is the one-time curation of the $4{,}743$ typed pairs from existing artifacts: it is real, it is human effort rather than API spend, and the data-scale curve (Figure 2b) is what prices it in accuracy terms. Figure 4 places routing accuracy against the cost axis: the specialist occupies the high-accuracy, no-marginal-cost corner, while the metered general embedder and the frontier LLM sit at higher cost and lower routing accuracy.

Integration proof-of-concept. To confirm the gain survives outside our own harness, we substitute an adapted encoder from our recipe into the real index-construction and routing code of a production LLM-as-judge evaluation framework, at its documented configuration seam, with no host-side changes; the test is an offline replay, not a live deployment. On the production catalog in place at the time of that test (21 skills, n=89 questions), the adapted encoder lifts Hit@1 from 0.517 (95% CI $[0.43, 0.63]$) for the general embedder to 0.742 ($[0.65, 0.83]$; Hit@3 0.888, Hit@5 0.921), and running that same encoder through the host's own index reproduces its standalone score to four decimal places, so the integration itself adds no distortion. The intervals do not overlap, so the gain is statistically separable and attributable to the embedder rather than the harness. Our headline 22M specialist, scored standalone on the same 89 questions, reaches 0.775 Hit@1, while a single-prompt frontier LLM routes at only 0.528. A per-skill analysis found that the few consistently-failing skills trace to stale gold labels (a newer, more specific skill now answers the question) or genuine sibling collisions that every embedder confuses, rather than to the adaptation, and no skill regressed from the general embedder to the adapted encoder.

6. Discussion and Limitations

Our results support a simple thesis: for the retrieval substrate of agentic e-commerce systems, the most cost-effective intervention is a small, domain-adapted encoder rather than a larger general model or a frontier LLM. Because the model is tiny, even full fine-tuning is cheap enough to repeat as the domain evolves, and the continual-learning recipe turns that into a routine weekly update that keeps the router current at roughly half the cost of retraining.

[1] Representative public list prices for commercial embedding APIs as of August 2026, cited as anchors for the cost tier rather than as any specific production vendor's terms.

[2] Public frontier-model list pricing as of August 2026, same caveat as above.

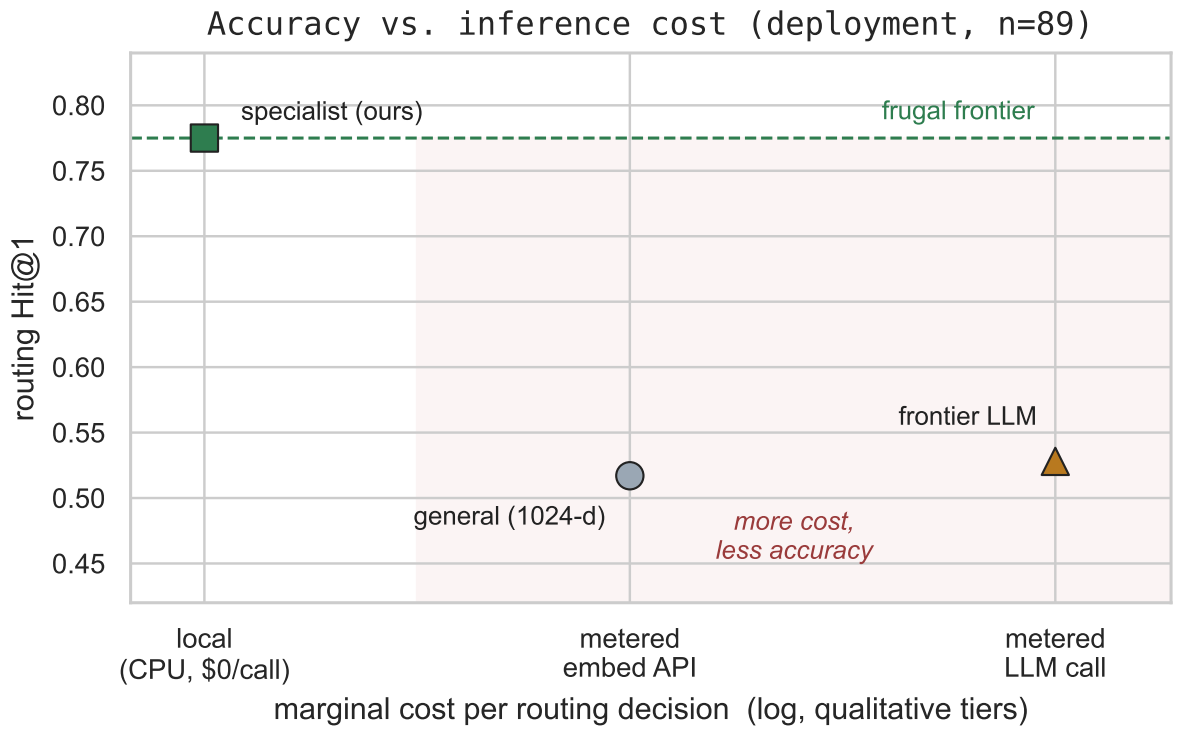


Figure 4: Frugality frontier: routing Hit@1 against the marginal cost of one routing decision. The horizontal axis is a log scale of three cost tiers ordered by operation type; tier positions, not exact prices, carry the argument, and each tier is anchored in the text by a measurement or a public list price: a local CPU forward pass (measured 8.1 ms median, zero marginal cost), a single metered embedding-API call (public list prices \$0.02–\$0.13 per million tokens), and a full autoregressive LLM decision ($\sim$2.8k prompt tokens, $\approx$\$0.014 at public list prices). The specialist sits at the top of the lowest-cost tier, higher in routing accuracy than the metered options while incurring no per-call cost.

Reconciling frugality with the unseen-skill gap. The clearest tension in our results is that the specialist wins in distribution but trails the general embedder on genuinely unseen skills, and that the fusion that fixes this reintroduces a per-query general-embedder call. We see the resolution as a *learned gate* rather than always-on fusion: a lightweight classifier, itself small, that decides per query whether the specialist can answer alone (the common in-distribution case, served locally at no marginal cost) or whether the query is novel or messy enough to justify a general-embedder call and a fused decision. This preserves the budget win on the majority of traffic and spends only where accuracy demands it. Building and evaluating that gate is our main line of future work, and it is the reason we report fusion as an upper-bound ablation rather than the headline.

Limitations. We state them plainly. *Scale and power*: the benchmarks are modest (n=308 for the static split, growing to 379 in the final round of the continual-learning evaluation), and the out-of-distribution arm in particular shifts with which skills are held out, so the robust signals are the per-model deltas on a fixed arm rather than any single absolute number. *Label provenance*: gold routes come from an LLM annotator that agrees with human labels at micro-F1 0.63 (exact) to 0.69 (rename-normalized), so absolute Hit@1 carries label noise of about that size; every model is scored against the identical gold, so the rankings are unaffected. Directionally, annotator error that is independent of the models deflates every arm's absolute Hit@1, since a correct route can be scored as wrong; inflation of a particular arm would require the annotator's errors to correlate with that model's errors. The plausible correlation runs through shared surface-lexical bias, which would, if anything, favor the general-purpose arms over the contrastively separated specialist, so we read the reported *deltas* as conservative. Multi-rater human gold is the durable fix and is on the future-work list. *Synthetic supplement*: some continual-learning

evaluation questions are model-generated for coverage of new and real-sparse skills, so we report the real-question arm separately and treat it as the production-critical metric. *Frontier-LLM comparison*: it uses a single fixed prompt, so it bounds rather than settles the model's capability. *Coverage and single-domain scope*: the measured coverage verdicts (Section 5.4) are threshold-dominated, so coverage thresholds are model-specific deployment configuration: they must be recalibrated per encoder rather than read as a fine-tuning effect, and replacing that manual recalibration with an automated procedure is itself future work. The evidence is one operational domain, so per-vertical gains remain to be measured even though the construction method is general.

Reproducibility scope. The training recipe is fully specified: the objective, optimizer, LoRA configuration, sequence length, learning rates, epochs, and temperature contain nothing proprietary. Under proprietary constraints we cannot release the corpus content or the identity of the production backbones, so the exact numbers are not independently replicable from this paper alone. We therefore release the dataset-construction methodology and sanitized exemplars; the recipe transfers to any in-domain corpus, and the specific figures should be read as evidence for the method rather than as a public benchmark.

7. Conclusion

We presented a frugal, domain-adaptive recipe for the skill-routing layer beneath agentic and LLM-as-judge e-commerce systems, in the two forms a deployed system needs. The static recipe constructs a leakage-free corpus of typed contrastive pairs and fully fine-tunes a small open-source encoder, lifting overall routing Hit@1 from 0.474 (a production general embedder) to 0.656 and in-distribution Hit@1 to 0.746 over a compact 384-d index at no marginal cost, a statistically separable gain delivered in a measured 8 ms median CPU decision, while trailing the general embedder only on genuinely unseen skills. The continual-learning recipe keeps that same small encoder current as the catalog grows from 23 to 43 skills, holding overall Hit@1 near 0.76 with warm-start plus a 30% replay buffer, matching a full retrain at about half the compute and avoiding catastrophic forgetting. Dropped into a production evaluator's real retrieval code with no host-side changes, an adapted encoder lifts routing Hit@1 from 0.517 to 0.742 over the general embedder, and the small specialist outperforms a single-prompt frontier LLM at zero marginal cost. For teams building agentic e-commerce systems, specializing the embedding layer is a practical and inexpensive improvement. Future work is a lightweight learned gate that captures the fusion upper bound while spending a general-embedder call only when a query needs it, together with multi-rater labels and a compact domain-aware judge model to complement the encoder.

Acknowledgments

We would like to thank the Automated Inventory Management (AIM) team in Amazon's Supply Chain Optimization Technologies (SCOT) organization for the production setting, the skill catalog, and the historical question registry this study builds on. We also thank the GenAIECommerce 2026 workshop reviewers for their constructive feedback, which greatly improved this camera-ready revision.

Declaration on Generative AI

During the preparation of this work, the author(s) used a generative AI assistant for grammar and spelling checks and for drafting assistance. After using these tools, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] M. Hejazinia, F. Balali, S. Rajabi, S. Jain, RADAR: LLM framework for root-cause analysis and data-driven automated reasoning at scale, 2025. In Proceedings of LLM4ECommerce Workshop at the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (LLM4ECommerce Workshop at KDD '25). ACM, New York, NY, USA.

[2] G. Nie, R. Zhi, X. Yan, Y. Du, X. Zhang, J. Chen, M. Zhou, H. Chen, T. Li, Z. Cheng, et al., A hybrid multi-agent conversational recommender system with llm and search engine in e-commerce, in: Proceedings of the 18th ACM Conference on Recommender Systems, 2024, pp. 745–747.

[3] H. Wang, X. Peng, H. Cheng, Y. Huang, M. Gong, C. Yang, Y. Liu, J. Lin, Ecom-bench: Can llm agent resolve real-world e-commerce customer support issues?, in: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, 2025, pp. 276–284.

[4] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., Judging llm-as-a-judge with mt-bench and chatbot arena, Advances in neural information processing systems 36 (2023) 46595–46623.

[5] N. Muennighoff, N. Tazi, L. Magne, N. Reimers, Mteb: Massive text embedding benchmark, in: Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, 2023, pp. 2014–2037.

[6] A. Allouah, O. Besbes, J. Figueroa, Y. Kanoria, A. Kumar, What is your AI agent buying? evaluation, implications, and emerging questions for agentic e-commerce, 2025. ArXiv preprint arXiv:2508.02630.

[7] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), 2019, pp. 3982–3992.

[8] T. Gao, X. Yao, D. Chen, Simcse: Simple contrastive learning of sentence embeddings, in: Proceedings of the 2021 conference on empirical methods in natural language processing, 2021, pp. 6894–6910.

[9] M. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, R. Kurzweil, Efficient natural language response suggestion for smart reply, arXiv preprint arXiv:1705.00652 (2017).

[10] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, Advances in neural information processing systems 33 (2020) 5776–5788.

[11] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, J.-Y. Nie, C-pack: Packed resources for general chinese embeddings, in: Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, 2024, pp. 641–649.

[12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, in: International Conference on Learning Representations (ICLR), 2022.

[13] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized llms, Advances in neural information processing systems 36 (2023) 10088–10115.

[14] K. Wang, N. Thakur, N. Reimers, I. Gurevych, Gpl: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval, in: Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies, 2022, pp. 2345–2360.

[15] J. Saad-Falcon, O. Khattab, K. Santhanam, R. Florian, M. Franz, S. Roukos, A. Sil, M. A. Sultan, C. Potts, Udapdr: Unsupervised domain adaptation via llm prompting and distillation of rerankers, in: Proceedings of the 2023 conference on empirical methods in natural language processing, 2023, pp. 11265–11279.

[16] M. McCloskey, N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of learning and motivation, volume 24, Elsevier, 1989, pp.

109–165.
[17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proceedings of the National Academy of Sciences 114 (2017) 3521–3526.
[18] A. Robins, Catastrophic forgetting, rehearsal and pseudorehearsal, Connection Science 7 (1995) 123–146.
[19] D. Lopez-Paz, M. Ranzato, Gradient episodic memory for continual learning, Advances in neural information processing systems 30 (2017) 6467–6476.
[20] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C. H. Lampert, icarl: Incremental classifier and representation learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2001–2010.