Targeting the Exceptions: Online Active Learning for Retrieval-Based LLM Routers

Clovis Varangot-Reille[1,2], Antoine Gourru[2], Christophe Bouvard[1], Baptiste Jeudy[2]
[1] Wikit, Lyon, France; [2] Laboratoire Hubert Curien, Université Jean Monnet Saint-Etienne, France
clovis.varangot@wikit.ai, antoine.gourru@univ-st-etienne.fr,
christophe.bouvard@wikit.ai, baptiste.jeudy@univ-st-etienne.fr

Abstract

Large language Models (LLM) routing reduces inference cost by selecting the most cost-effective model per query, but production deployment requires a router that adapts to an evolving model pool while adding negligible per-query overhead. Unsupervised retrieval-based routers satisfy both, but their support corpus must be built *online* from streaming traffic, and labeling each query requires running every candidate plus an LLM judge: prohibitively expensive at scale. We frame online corpus construction as stream-based active learning, exploiting a structural property of routing benchmarks: one or two models are globally near-optimal on the majority of queries, so the labeling budget should concentrate on the rare *exception regions* where selecting a specialist matters. We formalize this with a *performance sparsity* score, defined as the Kullback-Leibler divergence of a query's per-candidate performance vector from uniform: high when performance concentrates on a few models and low when all candidates are equivalent. Since this score depends on performances unavailable at inference time, we predict it online from query embeddings with a lightweight regressor. Across four benchmarks, our method SPARSENN approaches full-annotation routing performance with up to 44% fewer annotations in the stationary setting and 73% under topic shift. The results transfer to a fully held-out benchmark. A partial model annotation variant achieves higher cost savings, especially on large pools. Finally, both the corpus and the online estimator prove robust to model-pool changes.[1]

1 Introduction

As Large Language Model (LLM)-based applications scale to millions of daily queries, the up to 300× cost gap between frontier and lightweight LLMs has made query-level LLM selection a production necessity.[2] Trillion-parameter models are overkill for most queries, however, using only smaller ones might degrade user experience on complex tasks. *LLM routing* addresses this trade-off by selecting, for each query, the most cost-efficient candidate from a pool $\mathcal{M}$.

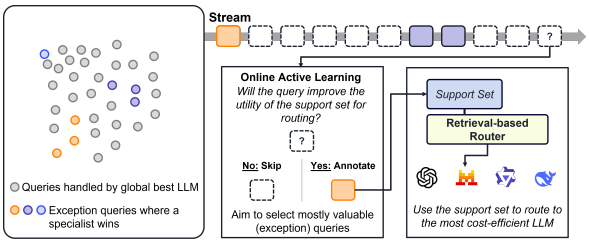


Figure 1: **Routing-aware active learning.** *Left:* Most queries (grey) are handled well by a single globally-strong default model; a small fraction (orange, blue, purple) form *exception regions* where different specialist models win. *Right:* As queries arrive in a stream, an online acquisition rule decides for each query whether annotating it would improve the support corpus used by the retrieval-based router. The budget is concentrated on exception queries. See Figure 2 for the full pipeline.

Production deployment imposes two constraints. First, $\mathcal{M}$ evolves rapidly as models are released and deprecated, ruling out routers that require retraining on every update. Second, the router runs on every query, so its own inference cost must remain marginal relative to the savings it delivers. These constraints favor *unsupervised retrieval-based routers* (Li, 2025; Jitkrittum et al., 2026; Varangot-Reille et al., 2026b), which estimate candidate performance by retrieving similar queries from a labeled support corpus. They require no parametric training, generalize easily to new candidates, and add minimal inference overhead. In production, however, queries arrive sequentially

[1] Code is available at: https://github.com/wikit-ai/online-active-learning-llm-routers.

[2] Based on output API pricing of GPT-5.5-Pro ($180.00 / 1M tokens) and Mistral-Small-4 ($0.60 / 1M tokens), as of May 4, 2026

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 52117–52135
October 24-29, 2026 ©2026 Association for Computational Linguistics

as an unbounded stream and the support corpus must be built *online*, and in cold-start settings (new deployments or distribution drift), entirely from scratch without historical labels. Each support entry pairs a query with the performance of *every* candidate in $\mathcal{M}$ on it, so annotating a single query requires generating a response from each of the $|\mathcal{M}|$ candidates and scoring every response with a high-capacity LLM judge. Annotating every query in a production stream is therefore prohibitive. We address this with an online active learning approach for retrieval-based routing that decides on the fly which queries can be informative. The aim is a strategy that builds the most informative support set while annotating as few queries as possible. Our contributions are:

- **A routing-aware view of active learning.** We show that LLM routing data has a *default/exception* structure: one or two models dominate on the vast majority of queries (top-two coverage $89-91\%$ on three of four benchmarks), while routing benefits come from a sparse set of exception queries where a specialist wins (Figure 1).
- **Performance sparsity as informativeness**: We propose KL-from-uniform of the per-query performance vector as an acquisition criterion that directly targets exception regions, and show that it can be inferred online via a lightweight regressor (Figure 2). Building on this criterion, SPARSENN matches full-annotation routing performance with up to 44% fewer annotations in the stationary setting and 73% under topic shift, outperforming random sampling and active-learning heuristics while staying consistently frontier-efficient in both regimes. These gains hold on a fully held-out benchmark. A partial-annotation variant further reduces total cost by up to 45% on large pools, and both the support set and the online estimator prove robust to model-pool changes at near-zero regret.

2 Related Work

LLM Routing Systems. LLM routers dynamically select, for each query, the most suitable model from a candidate pool $\mathcal{M}$, optimizing the trade-off between task performance and inference cost (Varangot-Reille et al., 2026a). They have gained strong industry traction, with production deployments at leading AI organizations[3], including OpenAI (Singh et al., 2026) and vLLM (Wang et al., 2025; Liu et al., 2026). In these settings, queries arrive as a stream and the pool $\mathcal{M}$ evolves continuously as models are released or deprecated. These are conditions under which existing router families differ sharply. Supervised routers train a classifier or regressor over query features to predict the best candidate (Chen et al., 2024; Feng et al., 2025a; Haque et al., 2025; Ong et al., 2025; Rida, 2026); while effective offline, they are tied to the pool seen at training time and require full retraining whenever $\mathcal{M}$ changes. Contextual bandit algorithms (Panda et al., 2025; Jafari and Farnia, 2026) update online from observed rewards but inherit the same coupling to a fixed candidate set. Fine-tuned LLM routers can in principle handle pool changes by re-prompting with an updated candidate (Zhang et al., 2026) or through intent detection (Tran et al., 2025), but their per-query inference cost contradicts the very savings routing is meant to deliver. Retrieval-based routers (Li, 2025; Jitkrittum et al., 2026; Varangot-Reille et al., 2026b) address both issues: they are model-agnostic, so adding or removing a candidate requires only updating the support corpus, and their per-query overhead is negligible. However, the performance of such routers is bounded by how informative the support corpus is. In this work, we study how to construct this corpus *online* under a finite annotation budget.

Active Learning. Active learning (AL) minimizes labelling cost by iteratively selecting the most informative unlabelled examples for annotation (Settles, 2009). The dominant *pool-based* formulation assumes a fixed, fully accessible, pool of unlabelled examples that can be scored at every acquisition round (Lewis and Gale, 1994; Sener and Savarese, 2018): an assumption that breaks down in production settings where queries arrive as an unbounded stream. *Stream-based* AL addresses this by deciding on the fly whether to annotate each incoming query, typically under a finite annotation budget (Aguiar and Cano, 2023; Cacciarelli and Kulahci, 2023). Acquisition strategies in both regimes are typically organized around two signals (Zhang et al., 2022): *informativeness*, most commonly framed as model uncertainty, which targets queries the current model is least confident about (Lewis and Gale, 1994) and remains the default

[3] See the HuggingFace Chat Omni Router, Azure AI Model Router, and NVIDIA LLM Router.

choice among NLP practitioners (Romberg et al., 2026); and *representativeness*, which seeks coverage of the input distribution (Sener and Savarese, 2018). Hybrid methods such as *BADGE* combine both via gradient embeddings (Ash et al., 2020). Whether they remain effective when the underlying task is routing has not, to our knowledge, been examined.

3 Problem Formulation

3.1 Query-Conditional LLM Routing

Let $\mathcal{M} = \{m_1, \ldots, m_n\}$ be a pool of LLMs and $\mathcal{Q}$ a query space. Each pair (q, m) has a performance value $s(q, m) \in \mathbb{R}_+$, e.g., exact-match accuracy on tasks with verifiable answers or a normalized LLM-as-a-judge quality score for open-ended generation. We write $\mathbf{s}_q = \big(s(q, m_1), \ldots, s(q, m_n)\big)$ the vector collecting the performance of all n candidates on q. For simplicity, we write $\mathbf{s}_i = \mathbf{s}_{q_i}$ throughout the remainder of the article. Each model m has an inference cost $C_m \in \mathbb{R}_+$. Given a cost-penalty parameter $\lambda \geq 0$, the optimal policy $\pi^* : \mathcal{Q} \to \mathcal{M}$ maximizes the per-query utility (Hu et al., 2024; Varangot-Reille et al., 2026a),

$$\pi^*(q) = \arg\max_{m \in \mathcal{M}} \big[s(q, m) - \lambda\, C_m\big]$$

Since s is unobserved at inference time, we substitute an estimator $\hat{s}$ and route via $\pi_{\hat{s}}$.

3.2 Retrieval-Based Routers.

Retrieval-based routers offer a generalizable, plug-and-play approach to performance estimation (Li, 2025; Jitkrittum et al., 2026; Varangot-Reille et al., 2026b). We implemented a k-nearest neighbor algorithm (k-NN) (Li, 2025; Varangot-Reille et al., 2026b), which rests on the assumption that semantically similar queries induce similar performance profiles across $\mathcal{M}$. Let

$$\mathcal{D}_{\text{sup}} = \{(q_i, \mathbf{s}_i)\}_{i=1}^{N},$$

be a labeled support set pairing each query with its ground-truth performance vector, and let $\mathcal{Q}_{\text{sup}} = \{q_i : (q_i, \mathbf{s}_i) \in \mathcal{D}_{\text{sup}}\}$ denote the set of queries within it. The support set must therefore be constructed so as to faithfully represent the partition structure of $\mathcal{Q}$. Queries are embedded via a pre-trained encoder $\phi : \mathcal{Q} \to \mathbb{R}^d$; for brevity, we hereafter conflate q with its embedding $\phi(q)$. For a query q, we retrieve the set $\mathcal{N}_k(q) \subseteq \mathcal{Q}_{\text{sup}}$ of its k nearest labeled queries under cosine similarity and estimate

$$\hat{s}_{k\text{-NN}}(q, m) = \frac{1}{k} \sum_{q' \in \mathcal{N}_k(q)} s(q', m)$$

Despite its simplicity, k-NN matches or outperforms parametric routers across several benchmarks (Li, 2025; Varangot-Reille et al., 2026b)

Online Corpus Construction as Stream-Based Active Learning. In production, queries arrive sequentially as a stream $\mathcal{S} = (q_t)_{t \geq 1}$, and $\mathcal{D}_{\text{sup}}$ must be built incrementally. Annotating q_t means obtaining its full performance vector $\mathbf{s}_t$, which requires generating and judging a response from each candidate. The per-query cost is the inference cost of all n candidates plus n LLM-judge calls. Since $|\mathcal{M}|$ can reach tens of models, annotating the whole stream is prohibitive, and *which* queries to annotate becomes a first-order concern, precisely what active learning addresses. We frame this as stream-based active learning under a budget $B \ll T$. At each step t, an acquisition function $\alpha_t : \mathcal{Q} \to \mathbb{R}$, depending only on q_t and $\mathcal{D}_{\text{sup}}^{(t-1)}$, scores the incoming query, and $a_t = \mathbb{1}[\alpha_t(q_t) \geq \tau_t]$ triggers annotation against an adaptive threshold τ_t. When $a_t = 1$, we pay the cost, obtain scores $\mathbf{s}_t$ and append $(q_t, \mathbf{s}_t)$ to $\mathcal{D}_{\text{sup}}$. We seek an acquisition policy $(\alpha_t, \tau_t)_{t=1}^{T}$ maximizing cumulative routing utility subject to the annotation budget:

$$\max_{(\alpha_t, \tau_t)} \sum_{t=1}^{T} \Big[s\big(q_t,\, \pi_{k\text{-NN}}^{(t)}(q_t)\big) - \lambda\, C_{\pi_{k\text{-NN}}^{(t)}(q_t)} \Big]$$
$$\text{s.t.} \sum_{t=1}^{T} a_t \leq B$$

where $\pi_{k\text{-NN}}^{(t)}(q_t)$ is the LLM selected for q_t by the k-NN router built from $\mathcal{D}_{\text{sup}}^{(t-1)}$. The next section introduces an acquisition function tailored to routing by exploiting the structure of $\mathbf{s}_q$ itself.

4 Performance Sparsity-Driven Active Acquisition

4.1 Performance Sparsity as Informativeness Score

The optimal policy π^* partitions $\mathcal{Q}$ into decision regions $Z_j = \{q : \pi^*(q) = m_j\}$, and the geometry of this partition governs annotation efficiency. On most benchmarks, $\mathcal{Q}$ collapses to a near-single region: the top-two candidates jointly attain best performance on 89-91% of queries on *RouterBench*,

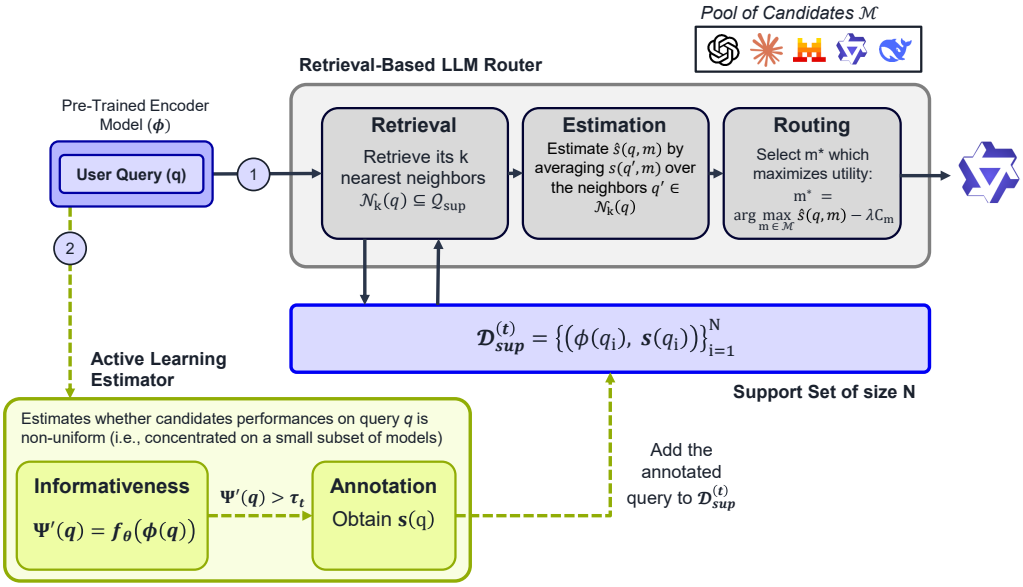


Figure 2: **Retrieval-based LLM routing pipeline with online active learning. (1) Retrieval-based LLM router** (top, grey): the k nearest neighbours in the support set are retrieved via cosine similarity; an estimator produces per-candidate scores; the router selects the candidate m^* which maximizes an utility score balanced via a cost penalty $\lambda \geq 0$. **(2) Active learning estimator** (bottom, yellow): a lightweight regressor $\Psi'(q) = f_\theta(\phi(q))$ estimates whether candidate performance on q is non-uniform (i.e., concentrated on a small subset of models); queries exceeding τ_t, the empirical β-percentile of Ψ values over the current support set, are annotated to obtain the per-candidate performance vector and appended to $\mathcal{D}_{\text{sup}}^{(t)}$; f_θ is retrained every δ annotations as $\mathcal{D}_{\text{sup}}$ grows.

FusionBench, and *Sprout* (Figure 11, Appendix M), making the routing task closer to best-arm identification. Routing thus decomposes into a globally strong *default* candidate covering most of the stream and rare *exception regions* where specialists outperform. The k-NN estimator recovers the default from few neighbors, so additional default-region labels are redundant; exception regions, by contrast, are the truly informative ones. At the other extreme, pools of similarly-sized and efficient models fragment $\mathcal{Q}$ into many regions: on *EmbedLLM*, whose $\sim$110 open-source models are of relatively uniform size (Zhuang et al., 2025), top-two coverage drops to 71.0%, so more queries carry routing-relevant signal. We now formalize the default/exception framework into an acquisition criterion. A labeled query $(q, \mathbf{s}_q)$ contributes to the support set by shifting the k-NN performance estimates in its neighborhood. When $\mathbf{s}_q$ is approximately uniform across candidates, this shift is non-directional: it does not favor one model over another, regardless of how many similar queries are retrieved alongside it. Informative queries are those for which $\mathbf{s}_q$ concentrates on a small subset of candidates, creating a clear routing preference in their neighborhood. We measure this concentration via the Kullback-Leibler (KL) divergence of the normalized performance profile, $\tilde{\mathbf{s}}_q$, from the uniform distribution over $\mathcal{M}$, where $\tilde{\mathbf{s}}_q$ is the ℓ_1-normalization of $\mathbf{s}_q$. The *informativeness* of q is:

$$\Psi(q) = D_{\text{KL}}\big(\tilde{\mathbf{s}}_q \,\big\|\, \mathcal{U}_{|\mathcal{M}|}\big) = \log|\mathcal{M}| - H\big(\tilde{\mathbf{s}}_q\big)$$

where H is the Shannon entropy. A peaked $\tilde{\mathbf{s}}_q$ where only a small subset of candidates attains high relative performance has high $\Psi(q)$. For a fixed pool $\mathcal{M}$, $\log|\mathcal{M}|$ is constant, so $\Psi(q)$ has ranking equivalence with $H\big(\tilde{\mathbf{s}}(q)\big)$. Even when $\mathcal{M}$ evolves, this equivalence holds within any single deployment, since queries are always compared under the same pool. We adopt $\Psi(q)$ rather than $-H(\mathbf{s}_q)$ as our criterion to preserve the natural "*larger is more informative*" orientation. Note that Ψ rewards concentration rather than superiority of a specific candidate; we discuss the resulting edge cases in the Limitations section.

Empirical validation. For each query q, we measure the *best-vs-default gap*: the performance difference between the per-query best candidate and the global best model. A useful acquisition signal concentrates budget on queries with large gaps.

Splitting each stream at the 75th percentile of Ψ, high-Ψ queries exhibit significantly larger best-vs-default gaps than low-Ψ queries on all four benchmarks at $p < 10^{-3}$, tested by a Mann-Whitney U test, with $r \in [0.16, 0.42]$ indicating small-to-moderate effect size (Peres, 2026) (Figure 13 in Appendix M).

4.2 Online Acquisition Algorithm

Selection rule. A query q_t is added to $\mathcal{D}_{\text{sup}}$ when $\Psi(q_t) > \tau_t$, where τ_t is the empirical β-percentile of Ψ values over $\mathcal{D}_{\text{sup}}^{(t)}$. Since $\mathbf{s}_t$ is unobserved at decision time, the rule is applied to a learned estimate $\Psi'(q_t)$.

Predicting informativeness from embeddings. We exploit a geometric regularity to estimate Ψ without observing $\mathbf{s}_t$: high-Ψ queries concentrate in localized regions of the embedding space (Figure 12 in Appendix M). We train a lightweight regressor $f_\theta : \mathbb{R}^d \to \mathbb{R}_+$ on $\mathcal{D}_{\text{sup}}$ ($\sim$ 10K parameters) producing $\Psi'(q) = f_\theta(\phi(q))$. The model f_θ is retrained periodically every δ annotations as $\mathcal{D}_{\text{sup}}$ grows (Figure 2). Architectural and training details are deferred to Appendix C.

Cold-start: covering the rule before targeting exceptions. Ψ-based acquisition is a refinement strategy. For the router to be efficient, $\mathcal{D}_{\text{sup}}$ must contain queries from the dominant region densely enough for the k-NN router to retrieve default-favoring neighbors. Relying on Ψ alone would invert this: an exception-heavy support set causes dominant-region queries to retrieve exception neighbors at inference and misroute to specialists. We therefore add a cold-start period of queries by annotating every incoming query until $|\mathcal{D}_{\text{sup}}| \geq N_{\text{cold}}$ to obtain unbiased coverage of $p(q)$ that establishes the dominant-region support. Once cold-start ends, f_θ is trained and the sparsity-driven rule takes over. The dominant region keeps growing, but only from queries that erroneously cross the threshold. We refer to this complete strategy, sparsity-driven acquisition with cold-start initialization, as SPARSENN.

Experimental Setup We evaluate each acquisition strategy on a stream $\mathcal{S}$ of length T. All strategies share the same k-NN routing procedure and the *snowflake-arctic-embed-m-v2.0* encoder (Yu et al., 2025) (0.3B parameters, 768 dimensions), differing only in their annotation policy. The details about the benchmarks used are described in Appendix E.

4.3 Metrics

At each step t, we route every query q from the test set $\mathcal{E}$ with the current $\mathcal{D}_{\text{sup}}^{(t)}$, retrieve the true performance and cost of the selected candidate, and average across $\mathcal{E}$ to obtain the per-step performance $\bar{P}_t$ and cost $\bar{C}_t$. From these we derive four metrics: **AUC of the performance curve** (AUC_P) is the area under $\bar{P}_{\tilde{t}}$; higher values indicate that strong routing performance is reached earlier and sustained throughout the stream. **AUC of the suboptimality curve** (AUC_{DTO}) is the area under the weighted distance-to-oracle in the cost-performance plane (Varangot-Reille et al., 2026b; Leteno et al., 2025); lower values indicate the router approaches the oracle earlier and stays close to it. Both reported over normalized time $\tilde{t} = t/T \in [0, 1]$. **Support set size** $|\mathcal{D}_{\text{sup}}^{(T)}|$ measures the total number of annotated queries at the end of the stream and is a proxy of the annotation cost. **Final performance** $\bar{P}_T$ reflects the routing quality of the support set ultimately constructed by each strategy.

4.4 Baselines

Each strategy is swept over a grid of acquisition thresholds. The values are listed in Appendix B.

Non-adaptive Sampling baselines. **Passive** annotates every incoming query ($a_t = 1$), giving full coverage of the stream until the budget is exhausted. **Random** annotates with $a_t \sim \text{Bernoulli}(\rho)$, providing unbiased coverage of p_t. Two oracles establish the ceiling for coverage-driven and informativeness-driven acquisition. The **Coverage Oracle** uses benchmark structure unavailable at inference: each benchmark consists of subdatasets, and given the K subdatasets $\{D_1, \ldots, D_K\}$ with empirical proportions ω_k, it allocates $b_k = \max(1, \lfloor \omega_k b \rfloor)$ samples per subdataset and draws them uniformly until the budget b is exhausted, isolating the value of optimal distributional coverage. The **Informativeness Oracle** has direct access to $\mathbf{s}_q$ at annotation time, computes the exact $\Psi(q)$, and applies the same threshold rule as our method, upper-bounding the exception-detection benefit on routing performance that any learned estimator f_θ can attain.

Traditional Active Learning Heuristics. We adapt the two traditional families of acquisition strategies, uncertainty and diversity (Settles, 2009),

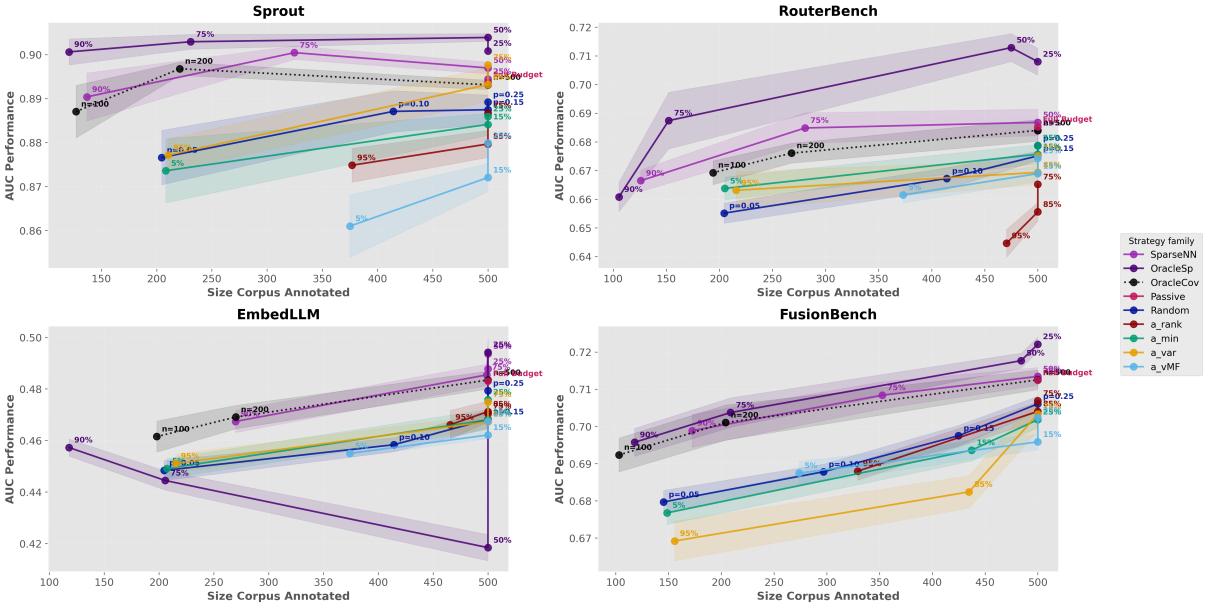


Figure 3: **SPARSENN sits at or near the frontier on all four benchmarks.** AUC_P vs. support corpus size, with each marker a distinct acquisition threshold. Points toward the upper-left, high performance at low annotation cost, are preferable. SPARSENN follows *OracleSp* on Sprout, RouterBench, and FusionBench, and follows the coverage oracle on EMBEDLLM, where no strong default exists. Cost penalty $\lambda = 0.25$; means over seeds (bands: standard error).

to routing. All share a common rule: at step t the query is annotated if its acquisition score exceeds τ_t, the β-percentile of a sliding window of the w most recent scores. **(a) Uncertainty heuristics** annotate queries where the router estimate is least stable near q. a_{var} scores q by the cosine-weighted variance of candidate performance over its k nearest neighbors in $\mathcal{D}_{\text{sup}}$, flagging regions where model performances fluctuate. a_{rank} instead targets *ranking* variability, a more decision relevant signal since routing depends on the argmax. It fits a Gaussian to each candidate's neighborhood performance, draws T Monte Carlo samples, and computes the average normalized Kendall-τ distance between the resulting rankings and the reference ranking induced by the weighted means. Both restrict the score to the top-m candidates, concentrating acquisition where the strongest models are hardest to separate rather than where weak models disagree. **(b) Diversity heuristics** broaden the coverage of $\mathcal{D}_{\text{sup}}$. a_{min} selects queries with low minimum cosine similarity to their k nearest neighbors. a_{vMF} targets queries in poorly covered regions of the embedding sphere via von Mises-Fisher (vMF) kernel density estimator (KDE) (Scott et al., 2021; Ju et al., 2025): low likelihood indicates a poorly covered region of the embedding sphere. Definitions are given in Appendix A.

5 Experiments

We evaluate acquisition strategies under two regimes (details in Appendix D): a **stationary** setting with a fixed query distribution, and a **non-stationary** setting where it shifts over time or the model pool evolves.

5.1 Stationary Setting

Queries are drawn i.i.d., so the decision partition $\{Z_1, \ldots, Z_J\}$ is constant. SPARSENN achieves the best or tied-best AUC_P on all four benchmarks while matching passive sampling at substantially fewer annotations (Figure 3). It reaches passive-sampling performance with 324.7 ± 15.7 annotations ($\beta = 75\%$) on Sprout (-35% vs. 500), 280.8 ± 28.3 ($\beta = 75\%$) on RouterBench (-44%), and 352.5 ± 22.5 ($\beta = 75\%$) on FusionBench (-30%). On EmbedLLM, it required the full budget to match passive sampling. At equal corpus size, SPARSENN is among the few strategies to achieve significantly better AUC_P and $\mathrm{AUC}_{\mathrm{DTO}}$ than random sampling on FusionBench and Sprout (Table 1), though not on EmbedLLM or RouterBench. It also shows significantly higher AUC_P and lower $\mathrm{AUC}_{\mathrm{DTO}}$ than the active-learning heuristics across all benchmarks, except the variance-based heuristic, which is statistically equivalent on all benchmarks except Em-

Metric	*OracleSp* (25%)	*OracleCov* (n=500)	SPARSENN (50%)	Passive	Rand (25%)	a_{rank} (75%)	a_{min} (25%)	a_{var} (75%)	a_{vMF} (25%)
Embedllm									
AUC Perf	49.4	48.3	**49.4**	48.3	47.9	47.0	47.6	47.5	46.7
AUC DTO	39.8	39.8	**39.0**	39.9	40.2	40.9	40.4	40.5	41.0
Final Perf	49.7	48.6	**49.8**	48.5	49.0	47.4	48.3	48.5	47.4
Fusionbench									
AUC Perf	72.2**	71.3	**71.3****	71.2	70.6	70.7	70.2	70.3	70.2
AUC DTO	16.9**	17.4**	**17.4****	17.4**	17.9	17.9	18.3	18.4	18.4
Final Perf	72.6**	71.5	**71.7**	71.5	71.3	71.3	71.2	71.5	70.9
Routerbench									
AUC Perf	70.8**	68.4	**68.7**	68.5	67.9	66.5**	67.9	67.5	67.4
AUC DTO	21.2	20.3	20.2	**20.2****	20.7	21.8**	21.0	21.2	21.2
Final Perf	71.3**	68.6	**68.9**	68.7	68.5	67.0**	68.4	68.1	67.9
Sprout									
AUC Perf	90.1**	89.3	89.7**	89.4	88.9	88.7	88.6	**89.8****	88.0**
AUC DTO	12.5**	10.9	12.1**	**10.9**	11.2	11.3	11.2	13.2**	11.5
Final Perf	90.2**	89.4	89.8	89.4	89.3	88.8	88.9	**90.0****	88.4
***R2-Bench** (Held-Out)*									
AUC Perf	75.6	-	75.6	75.4	**75.6**	75.5	75.4	75.5	75.5
AUC DTO	14.8	-	14.7	14.9	**14.7**	14.8	14.9	14.8	14.8
Final Perf	75.6	-	**75.7**	75.4	75.7	75.5	75.4	75.6	75.6

Table 1: **Stationary setting** ($k = 40$, Corpus Size = 500, 10 seeds). All methods are compared at a *fixed corpus size of 500* annotations; for each acquisition strategy we report the threshold variant β (shown in parentheses) reaching the highest AUC_P at that budget. This fixed-budget view is complementary to Figure 3, which reports the full annotation-efficiency frontier. Means over seeds; **bold**: best non-oracle per metric, underline: second best. For R2-Bench, *OracleCov* is not computed as the benchmark does not contain the original dataset of each query. ** marks strategies significantly different from Random (ρ =25%).

bedLLM, where SPARSENN remains significantly better (Table 2 in Appendix H). Our criterion is an effective measure of informativeness for routing: *OracleSp* dominates the AUC-vs-cost frontier on three of the four benchmarks, and SPARSENN follows it closely. On most benchmarks, coverage-driven acquisition is not the most efficient criterion (Figure 3). EmbedLLM is the exception. There, *OracleSp* underperforms every strategy except at its lowest threshold ($\beta = 25\%$): when performance is spread evenly across models (Section 4.1), aggressively selecting high-divergence queries excludes strong candidates that a broader corpus would cover. SPARSENN avoids this precisely because its estimator cannot recover the criterion exactly: Ψ is harder to predict when no dominant region anchors it, so prediction error inflates the annotation rate and restores coverage. It therefore diverge from *OracleSp* and follows the *OracleCov* instead. Finally, nearly all strategies show non-significant differences in final performance ($\bar{P}_T$) (Table 1). They are distinguished instead by annotation savings and how quickly performance rise matching the practical need for a strategy that is both fast to adapt and efficient.

Partial Model Annotation. Since performance concentrates on few models (Section 4.1) while annotation cost scales with pool size n, evaluating the full pool per query is largely redundant and wastes budget. For each selected query, we infer the performance of every candidate over its k nearest neighbors and compare each candidate against the best inferred one with a one-sided Wilcoxon signed-rank test. Only candidates not significantly worse than the best (those that could plausibly be top-1) are annotated. The rest have their performance imputed from the nearest-neighbor average. By redefining the budget as the *total annotation cost available* : the number of annotated queries $\times$ the cost of evaluating the full pool. The lower per-query cost lets more queries be labeled under the same B_{cost}. The benefit is pool-size dependent (Figure 7): on small pools ($|\mathcal{M}| \approx 10$; Sprout, RouterBench) few candidates can be safely pruned, so savings are negligible and AUC_P may slightly degrade. On larger pools it consistently helps: for SPARSENN ($\beta = 75\%$) it matches or exceeds full-pool AUC_P while saving 45% of the total cost on FusionBench ($|\mathcal{M}| \approx 40$) and 9% on EmbedLLM ($|\mathcal{M}| \approx 100$), where the full-annotation variant saves nothing. These results align with recent work questioning very large pools (Yuan et al., 2025).

5.2 Non-Stationary Setting

5.2.1 Topic Shift

Production query distributions drift as users ask about new topics. We simulate this by switching topics (e.g., health to mathematics) at fixed intervals through sudden drift (topic details in Appendix F). The partition $\{Z_j^{(t)}\}$ grows as previously unseen regions enter the stream. The results are more mixed than in the stationary setting (Figure 8 in Appendix J): SPARSENN remains frontier-efficient on all four benchmarks, whereas the uncertainty heuristic a_{var} reaches the frontier only on FusionBench and RouterBench. Passive sampling stays competitive despite exhausting its budget before later topics arrive, consistent with the structural property of Section 4.1: the default model selected in early phases continues to dominate later ones, so a corpus that never observes a topic still routes it adequately via the default. Most strategies match passive sampling's AUC_P at fewer annotations. For example, SPARSENN required 136.80 ± 9.3 annotations ($\beta = 90\%$) on Sprout (-73% vs. 500).

5.2.2 Pool Evolution.

Providers continually add and deprecate models. Since Ψ depends on the current pool, any change to $\mathcal{M}$ alters what the estimator must predict. We simulate this at the stream midpoint: starting from half the pool active, we randomly replace half of the active models with an equal number of inactive ones. Our strategy, *Continual*, trains on the old pool before the switch, re-fits online on the Ψ-relabeled corpus, and then continues training on post-switch queries. *Optimal* runs the full stream under the new pool from start, serving as an idealized reference. We measure *regret*, the *Continual*$-$*Optimal* gap on Performance and DTO: its value at the switch reflects how well the old corpus transfers, its decay how fast *Continual* adapts. Regret stays near zero throughout the post-switch stream on all four benchmarks (Figure 9 in Appendix K). At the switch, the corpus built under the old pool already matches the one *Optimal* builds from scratch on RouterBench, FusionBench, and Sprout. EmbedLLM is the only benchmark where it starts behind. Over the remainder of the stream, regret reaches zero on FusionBench and Sprout, while on EmbedLLM and RouterBench *Continual* ends ahead of *Optimal*. The old-pool corpus transfers useful information. If a gap remains at the switch, the estimator closes it online.

5.3 Sensitivity Analyses & Ablation

Hyperparameter robustness. We vary three hyperparameters on RouterBench, and analyze their impact on AUC_P. Performance rises with the *cold-start length* N_{cold} up to 75, then plateaus over the rest of the range $\{25, \ldots, 200\}$. Performance is not affected by the *retrain interval* $\delta \in \{1, \ldots, 50\}$, so f_θ does not need to be refreshed at every annotation, and larger δ reduces retraining overhead at no cost. For the *guardrail threshold* ϵ_{guard}, only the most conservative setting τ=0.10 impacted the performance, increasing corpus size without gain: the guardrail acts as a safety net for pathological f_θ failures, with likely larger impact on pools with more candidates. Figures are deferred to Appendix G.

Held-out generalization. To test results generalization, we re-ran the stationary experiment on R2-BENCH (Xue et al., 2026), a benchmark held out from all design and hyperparameter decisions (Table 1 and Figure 10 in Appendix L). At the matched 500-annotation budget, no strategy differs significantly from random sampling (Table 1) or SPARSENN (Table 2 in Appendix H), but the annotation-efficiency frontier reproduces the pattern seen previously: SPARSENN ($\beta = 90\%$) reaches comparable AUC_P with 72% fewer annotations (142 queries vs 500), staying frontier-efficient near *OracleSp*. We found the same default/exception spatial structure, and the top-two candidates cover 82% of queries.

6 Conclusion

Routing data exhibits a *default/exception* structure: a dominant default region carries most of the performance mass, while the specialist regions that remain are rare but locally coherent. We confirmed this across five benchmarks and showed that it leaves the traditional active-learning criteria, uncertainty and diversity, poorly aligned with the routing signal. We introduced *performance sparsity*, the KL divergence of the per-query performance vector from uniform, as a routing-aware criterion targeting specialist regions directly. Our method SPARSENN matches full-annotation routing with up to 73% fewer annotations under topic shift and 44% in the stationary setting, staying consistently frontier-efficient. Partial model annotation allows further cost savings at equal performance, especially on large candidate pools. Both corpus and online estimator prove robust to an evolving pool.

Limitations

Theoretical formalization. A formal guarantee would need to relate exception prevalence, acquisition precision and recall, routing margins, and k-NN coverage to routing regret. Establishing such guarantees is beyond the scope of this paper, and we leave it to future work.

Failure Modes of Ψ. The acquisition criterion Ψ rewards performance concentration, not success of a specific candidate, which introduces two known failure modes: **(i) Low-margin queries.** Queries with small but consistent inter-candidate gaps, e.g. $\mathbf{s}_q = (0.6, 0.5, 0.5)$, normalize to a near-uniform $\tilde{\mathbf{s}}(q)$ and receive low Ψ, even when a unique optimal candidate exists. Ψ will deprioritize these queries relative to higher-concentration ones, so the k-NN router may mis-route them at deployment time. **(ii) Default-dominated peaked queries.** Queries on which the globally dominant default is itself uniquely strong have highly peaked $\tilde{\mathbf{s}}(q)$ on the default and therefore high Ψ, even though annotating them carries no routing benefit: the k-NN router would select the default in their neighborhood regardless. This consumes budget that should be kept for true exception regions.

Distribution shift beyond topic switching. Our non-stationary experiments simulate abrupt drift via discrete topic switches. Real deployments also exhibit *gradual drift* (slow concept change), *incremental drift* (smooth interpolation between regimes), and *recurring drift* (seasonal or cyclic patterns). We expect Ψ-driven acquisition to remain effective under recurring drift, since the support corpus still contains samples from revisited regions, but its behavior under slow gradual or incremental drift is an open question we leave to future work.

References

Gabriel Aguiar and Alberto Cano. 2023. An active learning budget-based oversampling approach for partially labeled multi-class imbalanced data streams. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, SAC '23, page 382–389, New York, NY, USA. Association for Computing Machinery.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*.

Davide Cacciarelli and Murat Kulahci. 2023. Active learning for data streams: a survey. *Mach. Learn.*, 113(1):185–239.

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2025a. Graphrouter: A graph-based router for LLM selections. In *The Thirteenth International Conference on Learning Representations*.

Tao Feng, Haozhen Zhang, Zijie Lei, Pengrui Han, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Jiaxuan You. 2025b. Fusionfactory: Fusing LLM capabilities with multi-LLM log data. *Preprint*, arXiv:2507.10540.

Eduardo García-Portugués. 2013. Exact risk improvement of bandwidth selectors for kernel density estimation with directional data. *Electronic Journal of Statistics*, 7:1655–1685.

Mirazul Haque, Petr Babkin, Vali Tawosi, Saba Rahimi, Natraj Raman, Zhiqiang Ma, and Xiaomo Liu. 2025. Codyn: Dynamic LLM routing for coding tasks. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*.

Donya Jafari and Farzan Farnia. 2026. DAK-UCB: Diversity-aware prompt routing for LLMs and generative models. In *The Fourteenth International Conference on Learning Representations*.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. 2026. Universal model routing for efficient LLM inference. In *The Fourteenth International Conference on Learning Representations*.

Li Ju, Max Andersson, Stina Fredriksson, Edward Glöckner, Andreas Hellander, Ekta Vats, and Prashant Singh. 2025. Exploiting the asymmetric uncertainty structure of pre-trained VLMs on the unit hypersphere. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Thibaud Leteno, Michael Perrot, Charlotte Laclau, Antoine Gourru, and Christophe Gravier. 2025. Fair text classification via transferable representations. *Journal of Machine Learning Research*, 26(239):1–47.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR '94*, pages 3–12, London. Springer London.

Yang Li. 2025. Rethinking predictive modeling for LLM routing: When simple kNN beats complex learned routers. *Preprint*, arXiv:2505.12601.

Xunzhuo Liu, Huamin Chen, Samzong Lu, Yossi Ovadia, Guohong Wen, Hao Wu, Zhengda Tan, Jintao Zhang, Senan Zedan, Yehudit Kerido, Liav Weiss, Haichen Zhang, Bishen Yu, Asaad Balum, Noa Limoy, Abdallah Samara, Baofa Fan, Brent Salisbury, Ryan Cook, and 11 others. 2026. vLLM semantic router: Signal driven decision routing for mixture-of-modality models. *Preprint*, arXiv:2603.04444.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. 2025. Adaptive LLM routing under budget constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23934–23949, Suzhou, China. Association for Computational Linguistics.

Fernanda Peres. 2026. Effect sizes for nonparametric tests. *Biochemia medica*, 36:5–16.

Adam Rida. 2026. TRACER: Trace-based adaptive cost-efficient routing for LLM classification. *Preprint*, arXiv:2604.14531.

Julia Romberg, Christopher Schröder, Julius Gonsior, Katrin Tomanek, and Fredrik Olsson. 2026. Reassessing active learning adoption in contemporary NLP: A community survey. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2621–2647, Rabat, Morocco. Association for Computational Linguistics.

Tyler R. Scott, Andrew C. Gallagher, and Michael C. Mozer. 2021. von Mises–Fisher loss: An exploration of embedding geometries for supervised learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10592–10602.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, and 467 others. 2026. OpenAI GPT-5 system card. *Preprint*, arXiv:2601.03267.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. 2025. CARROT: A cost aware rate optimal router. *Preprint*, arXiv:2502.03261.

Co Tran, Salman Paracha, Adil Hafeez, and Shuguang Chen. 2025. Arch-router: Aligning LLM routing with human preferences. *Preprint*, arXiv:2506.16655.

Clovis Varangot-Reille, Christophe Bouvard, Mathieu Ciancone, Antoine Gourru, Marion Schaeffer, and François Jacquenet. 2026a. Doing More with Less: A survey on routing strategies for resource optimisation in large language model-based systems. *Journal of Artificial Intelligence Research*, 86.

Clovis Varangot-Reille, Christophe Bouvard, and Antoine Gourru. 2026b. Generalising LLM routing using past performance retrieval: A few-shot router is sufficient. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 304–319, Rabat, Morocco. Association for Computational Linguistics.

Chen Wang, Xunzhuo Liu, Yuhan Liu, Yue Zhu, Xiangxi Mo, Junchen Jiang, and Huamin Chen. 2025. When to Reason: Semantic router for vLLM. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems: Workshop on Machine Learning for Systems*.

Jiaqi Xue, Qian Lou, Jiarong Xing, and Heng Huang. 2026. R2-router: A new paradigm for LLM routing with reasoning. In *Forty-third International Conference on Machine Learning*.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel F Campos. 2025. Arctic-embed 2.0: Multilingual retrieval without compromise. In *Second Conference on Language Modeling*.

Jiayi Yuan, Yifan Lu, Rixin Liu, Yu-Neng Chuang, Hongyi Liu, Shaochen Zhong, Yang Sui, Guanchu Wang, Jiarong Xing, and Xia Hu. 2025. Who routes the router: Rethinking the evaluation of LLM routing systems. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.

Haozhen Zhang, Tao Feng, and Jiaxuan You. 2026. Router-R1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zhisong Zhang, Emma Strubell, and Eduard Hovy. 2022. A survey of active learning for natural language processing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6166–6190, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. 2025. EmbedLLM: Learning compact representations of large language models. In *The Thirteenth International Conference on Learning Representations*.

A Heuristic Acquisition Functions

This section defines the four acquisition scores. The threshold τ_t is set to a high percentile of the sliding window for a_{var} and a_{rank} (higher scores indicate higher uncertainty) and to a low percentile for $a_{\min}$ and a_{vMF} (lower scores indicate poorly covered regions). The swept values are detailed in Appendix B.

A.1 Uncertainty Heuristics

Let $\mathcal{N}_k(q)$ denote the k nearest neighbors of q in $\mathcal{D}_{\text{sup}}$ under cosine similarity, with weights $w_j \propto \cos(q, q_j)$ normalized to sum to one, and let $\mu_w(q, m_i)$ and $\sigma_w^2(q, m_i)$ denote the weighted mean and weighted variance of candidate m_i's performance over $\mathcal{N}_k(q)$.

Weighted variance.

$$a_{\text{var}}(q) = \frac{1}{n}\sum_{i=1}^{n} \sigma_w^2(q, m_i)$$

Kendall-τ ranking instability. The reference ranking ξ^* is obtained by sorting candidates by $\mu_w(q, m_i)$, and at each iteration $\ell \in \{1, \dots, T\}$ we sample a performance score for each model from $\mathcal{N}\big(\mu_w(q, m_i),\, \sigma_w^2(q, m_i)\big)$ and re-rank to obtain $\xi^{(\ell)}$; with $\text{disc}(\xi, \xi^*)$ counting discordant pairs over all ordered index pairs (i, j), $i \neq j$. The normalized Kendall-τ distance is

$$\bar{d}_K(\xi, \xi^*) = \frac{\text{disc}(\xi,\, \xi^*)}{n(n-1)}$$

and the ranking uncertainty score is

$$a_{\text{rank}}(q) = \frac{1}{T}\sum_{\ell=1}^{T} \bar{d}_K\Big(\xi^{(\ell)}, \xi^*\Big)$$

A.2 Diversity Heuristics

Minimum neighbor similarity.

$$a_{\min}(q) = \min_{q' \in \mathcal{N}_k(q)} \cos(q,\, q')$$

von Mises-Fisher (vMF) Kernel Density Estimation (KDE). For ℓ_2-normalized embeddings on $\mathbb{S}^{d-1}$, where d is the embedding dimension, the vMF density(García-Portugués, 2013) is

$$f(x\,;\,\mu, \kappa) = C_d(\kappa)\, \exp\Big(\kappa\, \mu^\top x\Big),$$
$$C_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2}\, I_{d/2-1}(\kappa)} \tag{1}$$

with $I_\nu(\cdot)$ the modified Bessel function of order $\nu = d/2 - 1$. For high-dimensional embeddings $I_{d/2-1}(\kappa)$ is numerically intractable; following Scott et al. (2021); Ju et al. (2025) we use the approximation:

$$F_d(\kappa) = -\kappa + \frac{d-1}{2}\ln\kappa - \frac{d-1}{2}\ln(2\pi)$$

a numerically stable approximation to $\ln C_d(\kappa)$ for any d. Given a corpus of n embeddings $\{x_i\}_{i=1}^n \subset \mathbb{S}^{d-1}$ and a query $q \in \mathbb{S}^{d-1}$, with log-sum-exp stabilization $m = \max_i(\kappa, x_i^\top q)$, the diversity score becomes the KDE log-likelihood:

$$a_{\text{vMF}}(q) = F_d(\kappa) - \ln n + m + \ln\sum_{i=1}^{n} \exp\Big(\kappa\, x_i^\top q - m\Big) \tag{2}$$

B Threshold Sweep Configuration

We sweep the acquisition threshold or sampling rate of every method. SPARSENN and *OracleSp* use $\{0.25, 0.50, 0.75, 0.90\}$. Random uses $\{0.05, 0.10, 0.15, 0.25\}$. For the uncertainty heuristics a_{var} and a_{rank}, both with top-k $= 3$, we sweep $\{0.75, 0.85, 0.95\}$. For the diversity heuristics $a_{\min}$ and a_{vMF} with $\kappa = 1$, we sweep $\{0.05, 0.15, 0.25\}$. For *OracleCov*, which depends on a target corpus size, we use $b \in \{100, 200, 500\}$.

C Training Details

Architecture. The network consists of two hidden layers of dimension d_h, each followed by ReLU activation and dropout ($p = 0.2$), and a final linear layer projected to a scalar through a Softplus activation, enforcing non-negativity consistent with $\Psi(q) \geq 0$.

Cold start. No training is performed until $N_{\text{cold}} = 115$ labeled samples have been collected. During this phase, all the incoming queries are annotated.

Training protocol. The model is not truly online: at each trigger (every δ new labeled samples), it is retrained from scratch on the *full annotated corpus* accumulated so far. This prevents catastrophic forgetting. At inference, the acquisition threshold τ_t is set to the empirical β-percentile of Ψ' values over the current support set. The swept values of β are given in Appendix B.

Hyperparameters Optimization. We have selected the hyperparameters $(\eta, w, d_{\text{hidden}})$ via grid search over $\{10^{-4}, 10^{-3}\} \times \{10^{-3}, 10^{-2}\} \times \{16, 32, 64, 128\}$ on a held-out validation split (20%) drawn from the stationary training stream. The optimal values were found to be: $\eta = 10^{-3}$, $w = 10^{-3}$, $d_{\text{hidden}} = 16$.

Optimization. During training, we minimize MSE with AdamW for up to 50 epochs, restoring weights to the best validation checkpoint upon early stopping (patience $= 10$).

Validation, early stopping, and guardrail.

- A held-out set (15% of $\mathcal{D}_{\text{sup}}$) is drawn from older observations, with the δ newest points forced into the training split. This is *not a true generalization estimate*: since the model is retrained from scratch at each trigger, the held-out points have been seen in earlier rounds. The validation loss serves as a *stability regularizer* via early stopping (patience $= 10$ epochs). It prevents the model from overfitting the most recent batch at the expense of previously acquired knowledge.
- The final validation MSE acts as a *validation quality guardrail*: whenever it exceeds a threshold $\epsilon_{\text{guard}} = 0.20$, the model is considered unreliable and every subsequent item is selected unconditionally, regardless of its predicted KL divergence. Each forced selection appends the observation to the learner's dataset; a full retraining round is triggered only once every δ new observations have accumulated.

D Experiment Details

Results are averaged over $R = 10$ seeds, with the stream $\mathcal{S}^{(r)}$ and test set $\mathcal{E}^{(r)}$ resampled at each seed and all strategies re-run on the same $\mathcal{S}^{(r)}$. We fix an annotation budget of $B = 500$, a cold-start length of $N_{\text{cold}} = 115$, a retrain interval of δ =5, and a cost penalty of $\lambda = 0.25$.

D.1 Stationary Details

We sample 4000 queries for the stream and 5000 for the test set, with k =40. For FusionBench, which is smaller, we preserve the same $44.4/55.6\%$ stream/test proportion.

D.2 Partial Annotation Details

The pairwise tests are corrected for multiple comparisons via Holm-Bonferroni at $\alpha = 0.05$.

D.3 Topic Shift Details

The stream is partitioned into sequential topic phases with a sudden switch at each boundary, drawing 500 queries per topic for both the stream and the test set. The number of topics is 3 for Sprout, 4 for RouterBench, 2 for FusionBench, and 4 for EmbedLLM; topic definitions and ordering are given in Appendix F.

D.4 Pool Evolution Details

We sample 3000 queries for the stream and 5000 for the test set, with k =40. The stream is split in two equal halves. During the first half, the acquisition strategy operates on a random subset of $\lfloor M/2 \rfloor$ active models out of the full pool of M models. At the midpoint $t = 1500$, we simulate a pool change: $\lfloor M/4 \rfloor$ active models are removed and $\lfloor M/4 \rfloor$ previously inactive models are added, with both sets chosen uniformly at random. For EmbedLLM, which has a higher selection rate, we raise the budget to $B = 1000$ to prevent exhaustion before the switch.

E Benchmark Details

SPROUT-o3mini (Somerstep et al., 2025). A benchmark of ∼30k queries across 14 candidate models, spanning open-weight models ranging from 1B to 70B parameters (LLaMA-3.1 1B-70B; Granite-3 2B and 8B; Mixtral-8x7B), alongside proprietary systems of undisclosed scale (Claude-3.5-Sonnet, GPT-4o, GPT-4o-mini, o3-mini). Tasks are drawn from GPQA, MuSR, MMLU-Pro, MATH, Teknium, and RAG-Bench, collectively probing graduate-level reasoning, mathematical problem-solving, and retrieval-augmented generation.

RouterBench (Hu et al., 2024). A benchmark of ∼36k queries across 11 candidate models, including proprietary APIs (GPT-3.5-turbo, GPT-4, Claude Instant, Claude v1/v2) and

open-weight models (LLaMA-2-70B, CodeLlama-34B, Mistral-7B, Mixtral-8x7B, WizardLM-13B, Yi-34B). Tasks span coding (MBPP), mathematical reasoning (GSM8K), commonsense inference (HELLASWAG, WINOGRANDE, ARC-CHALLENGE), factual knowledge (MMLU), and open-ended generation (MTBENCH). To improve interpretability and remove any linguistic impact on the embeddings, we did not include the Chinese datasets.

FusionBench (Feng et al., 2025b). A benchmark of ∼6k queries evaluating 20 open-weight base models ranging from 7B to 122B parameters (Gemma, Granite, LLaMA-3.1/3.3, Mistral, Mixtral, Qwen2/2.5, and others) each under two inference settings: standard and chain-of-thought, for a total of 40 candidate configurations. Tasks cover reasoning (ARC-CHALLENGE, BOOLQ, HELLASWAG), mathematics (MATH, GSM8K), coding (HUMANEVAL, MBPP), and question answering (TRIVIAQA, NATURALQA, SQUAD).

EmbedLLM (Zhuang et al., 2025). : A benchmark of ∼30k queries spanning 108 candidate models (originally 112, but 4 candidates were removed as we could not infer their number of parameters). Model sizes range from 1.3B to 70B parameters, with 7B and 13B models being the most prevalent, and the pool includes a large proportion of domain-specialized fine-tunes and model merges across health, mathematics, and code. Tasks span mathematical reasoning (GSM8K, ASDIV, MATHQA), medical knowledge (MEDMCQA), commonsense inference (PIQA, LOGIQA, SOCIALIQA), factual knowledge (MMLU, GPQA), and factuality (TRUTHFULQA).

R2-Bench (Xue et al., 2026). A benchmark of ∼31k queries spanning 10 candidate models and held out from all design and hyperparameter decisions to test the generalizability of our findings. Model sizes range from 0.6B to 235B parameters, mixing lightweight open-weight models (Qwen3-0.6B, LLaMA-3.2-3B) with math-specialized fine-tunes (Qwen2.5-Math-1.5B/7B) and larger generalist ones (GLM-4.5-Air, GLM-4.6, Qwen3-235B-A22B). On the 21/07/2026, Gemma-3-1B, Gemma-3-4B-it, and Mistral-7B performances are not available on the HuggingFace repository. If found, cost were updated on 21/07/2026 based on openrouter.ai, otherwise, we used the original cost reported in the paper. Tasks span professional-level knowledge (MMLU-PRO), general knowledge and reasoning (OPENHERMES), mathematics (MATH), graduate-level science (GPQA), multi-step soft reasoning (MUSR), and retrieval-augmented generation tasks (RAGBENCH). We have selected the subset of 500 tokens budget because of its higher compliance rate among models (Xue et al., 2026).

F Topics for Non-Stationary Evaluation

We define topics to simulate non-stationarity. Each topic is a sequential phase of the stream.

SPROUT-o3mini (3 topics). Mathematics: MATH. **RAG & Task Reasoning**: RAGBENCH (CovidQA, PubMedQA, HotpotQA, MS-MARCO, HAGRID, ExpertQA, DelucionQA, CUAD, eManual, FinQA, TAT-QA, TechQA) and MUSR (Object Placements, Team Allocation). **Question Answering**: MMLU-PRO.

RouterBench (4 topics). Mathematics: GSM8K, MTBENCH-Math, and MMLU math subsets. **Science**: MMLU physics, chemistry, biology, astronomy, and electrical engineering subsets. **Medicine & Health**: MMLU medical subsets. **Law & Ethics**: MMLU legal and moral subsets and BIAS DETECTION.

FusionBench (2 topics). Reasoning: MATH, GSM8K, ARC-CHALLENGE, HELLASWAG, MBPP, HUMANEVAL. **Question Answering**: GPQA, TRIVIAQA, NATURALQA, SQUAD, OPENBOOKQA, COMMONSENSEQA, BOOLQ, MMLU.

EmbedLLM (4 topics). Medicine & Health: MEDMCQA and MMLU medical subsets. **Mathematics**: MATHQA, ASDIV, GSM8K, and MMLU math subsets. **Social Sciences**: SOCIALIQA and MMLU social subsets. **Science**: GPQA and MMLU science subsets.

G Sensitivity and Ablation Analyses Figures (RouterBench)

Error bars show standard error across seeds.

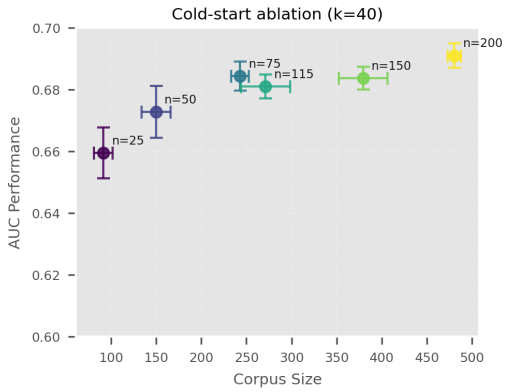


Figure 4: Ablation over cold-start sizes.

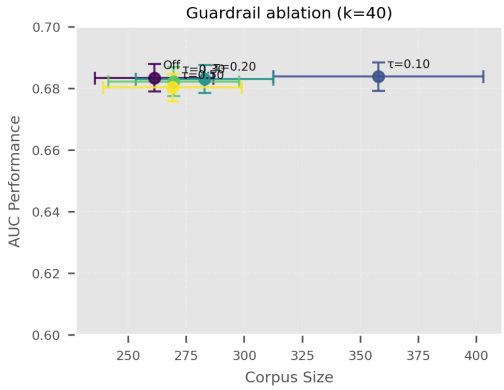


Figure 5: Ablation over guardrail thresholds τ. *Off*: no guardrail applied.

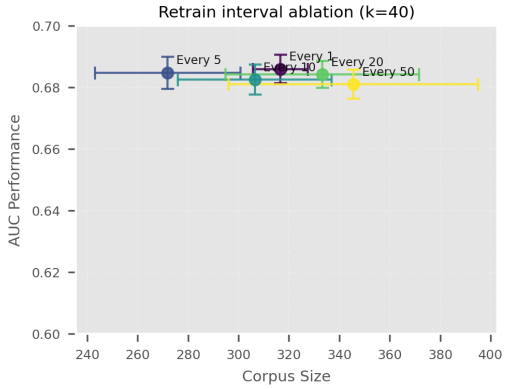


Figure 6: Ablation over retraining intervals.

H Significancy Tests against SPARSENN

Metric	*OracleSp* (25%)	*OracleCov* (n=500)	SPARSENN (50%)	Passive	Rand (25%)	a_{rank} (75%)	a_{min} (25%)	a_{var} (75%)	a_{vMF} (25%)
EmbedLLM									
AUC Perf	49.4	48.3	**49.4**	48.3	47.9	47.0**	47.6**	47.5**	46.7**
AUC DTO	39.8	39.8	**39.0**	39.9	40.2	40.9**	40.4**	40.5**	41.0**
Final Perf	49.7	48.6	**49.8**	48.5	49.0	47.4	48.3	48.5	47.4**
Fusionbench									
AUC Perf	72.2**	71.3	**71.3**	71.2	70.6**	70.7	70.2**	70.3	70.2**
AUC DTO	16.9**	17.4	**17.4**	17.4	17.9**	17.9	18.3**	18.4	18.4**
Final Perf	72.6**	71.5	**71.7**	71.5	71.3	71.3	71.2	71.5	70.9
Routerbench									
AUC Perf	70.8**	68.4	**68.7**	68.5	67.9	66.5**	67.9	67.5	67.4**
AUC DTO	21.2	20.3	20.2	**20.2**	20.7	21.8**	21.0**	21.2	21.2**
Final Perf	71.3**	68.6	**68.9**	68.7	68.5	67.0**	68.4	68.1	67.9
Sprout									
AUC Perf	90.1	89.3	89.7	89.4	88.9**	88.7**	88.6	**89.8**	88.0**
AUC DTO	12.5	10.9**	12.1	**10.9****	11.2**	11.3**	11.2**	13.2**	11.5
Final Perf	90.2	89.4	89.8	89.4	89.3	88.8**	88.9	**90.0**	88.4**
***R2-Bench** (Held-Out)*									
AUC Perf	75.6	-	75.6	75.4	**75.6**	75.5	75.4	75.5	75.5
AUC DTO	14.8	-	14.7	14.9	**14.7**	14.8	14.9	14.8	14.8
Final Perf	75.6	-	**75.7**	75.4	75.7	75.5	75.4	75.6	75.6

Table 2: **Stationary setting** ($k = 40$, Corpus Size = 500, 10 seeds). All methods are compared at a *fixed corpus size of 500* annotations; for each acquisition strategy we report the threshold variant β (shown in parentheses) reaching the highest AUC_P at that budget. This fixed-budget view is complementary to Figure 3, which reports the full annotation-efficiency frontier. Means over seeds; **bold**: best non-oracle per metric, underline: second best. For R2-Bench, *OracleCov* is not computed as the benchmark does not contain the original dataset of each query. ** marks strategies significantly different from SPARSENN.

I Partial Model Annotation

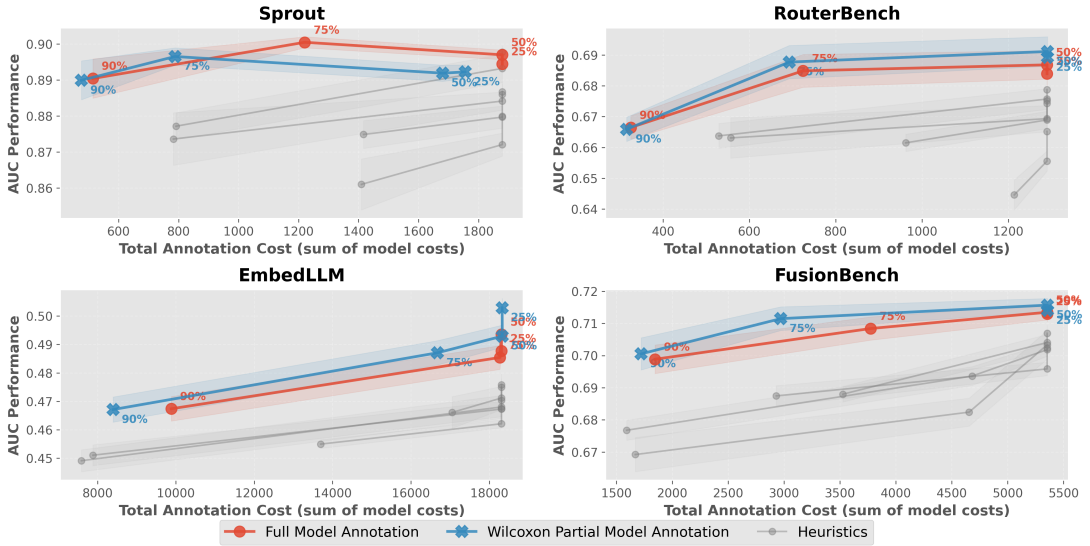


Figure 7: **Partial model annotation pays off on large pools, where candidates performance are most redundant. Full model annotation** (red) evaluates every candidate on each selected query; **Wilcoxon partial annotation** (blue) skips candidates significantly below the inferred neighborhood best candidate (one-sided Wilcoxon signed-rank test, α=0.05, Holm correction). Both operate under the same total-cost budget, so cheaper per-query annotation allows partial annotation label more queries. The benefit is pool-size dependent: negligible on small pools (Sprout, RouterBench) but significant once the pool is large enough (FusionBench, EmbedLLM). Grey lines show heuristic baselines. Means over seeds (bands: standard error).

J Annotation-Efficiency Under Topic Shift

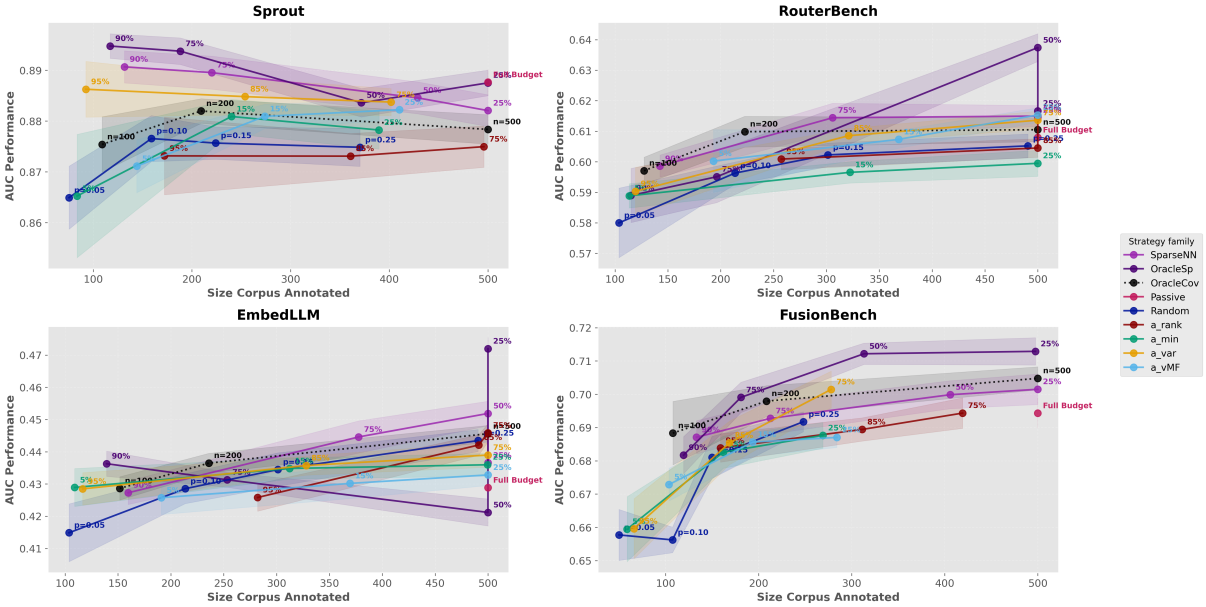


Figure 8: **Under sudden topic shift, most strategies overlap, but SPARSENN is consistently among the frontier-efficient ones on all four benchmarks.** Individual heuristics also reach the frontier on some benchmarks but none does so consistently across all four, except a_{var}. On EMBEDLLM, where no strong default exists, *OracleSp* degrades at aggressive thresholds and SPARSENN instead follows the coverage oracle. Means over seeds (bands: standard error).

K Regret After Pool Evolution

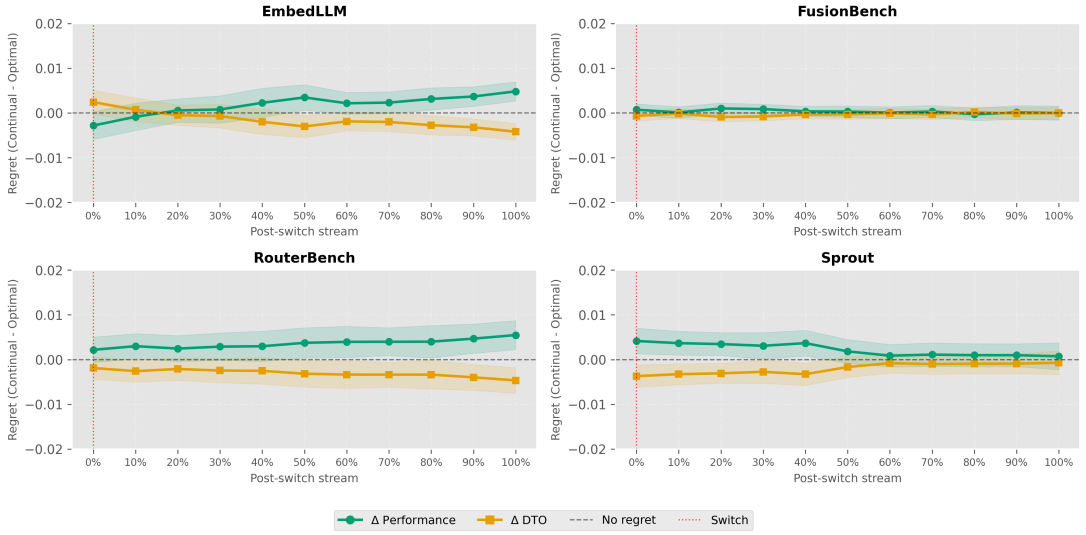


Figure 9: **A corpus built for a different pool still transfers, and SPARSENN's estimator adapts online: regret after a model-pool switch stays near zero throughout.** Regret (*Continual*−*Optimal*) on Performance (green) and DTO (yellow) through the post-switch stream, where the pool change happens at 0% (dotted red line). Performance is higher-is-better and DTO lower-is-better, so *Continual* underperforms *Optimal* for negative green values and positive yellow ones. At the switch, regret is near zero on RouterBench, FusionBench, and Sprout, and only EmbedLLM starts behind. Over the stream, regret reaches zero on FusionBench and Sprout, while *Continual* ends ahead of *Optimal* on EmbedLLM and RouterBench. Means over seeds (bands: standard error).

L Held-Out Generalization: R2-Bench

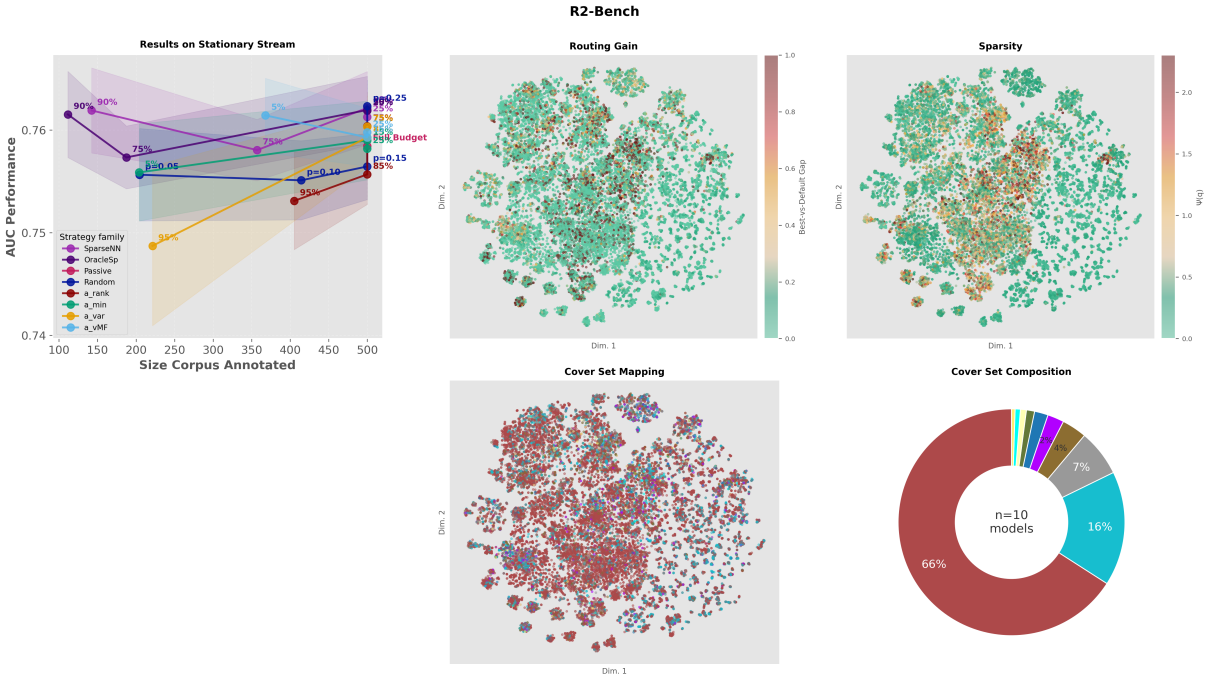


Figure 10: **Held-out R2-BENCH confirms the default/exception structure.** (a) Annotation-efficiency frontier (AUC_P vs. corpus size): SPARSENN follows *OracleSp* and reaches full-budget performance with far fewer annotations. Means over seeds (bands: standard error). (b) Best-vs-default gap and (c) Ψ, both on a t-SNE projection of query embeddings: routing gains concentrate in spatially coherent exception regions. (d) and (e) Cover-set composition: the top-two candidates jointly cover 82% of queries, confirming a dominant default.

M Empirical Validation Figures

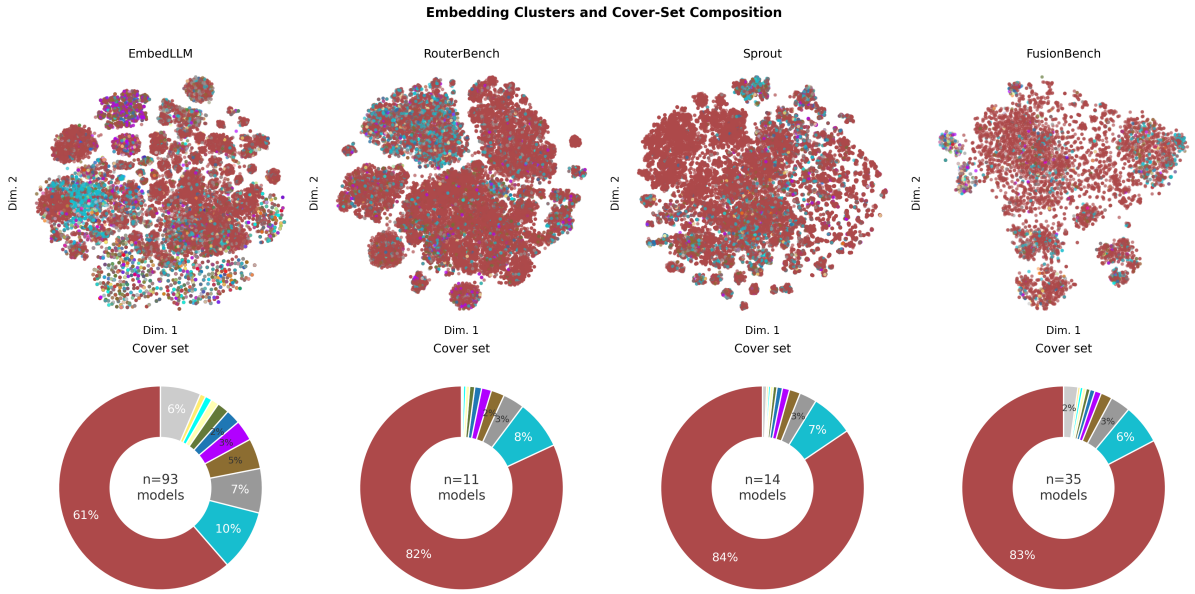


Figure 11: **Two models jointly cover 89–91% of queries on three of four benchmarks, confirming a dominant default with exception regions.** Each column is one benchmark. *Top*: t-SNE projection of query embeddings colored by the model assigned under a greedy set-cover algorithm, which iteratively selects the model covering the most yet-uncovered queries (a query is covered if a model's score is within tolerance $\varepsilon = 0.05$ of its per-query best). Spatial coherence of colored regions shows that exception queries cluster in the embedding space, motivating embedding-based sparsity prediction. *Bottom*: Donut chart of coverage fractions; the center reports the total number of models required for full coverage.

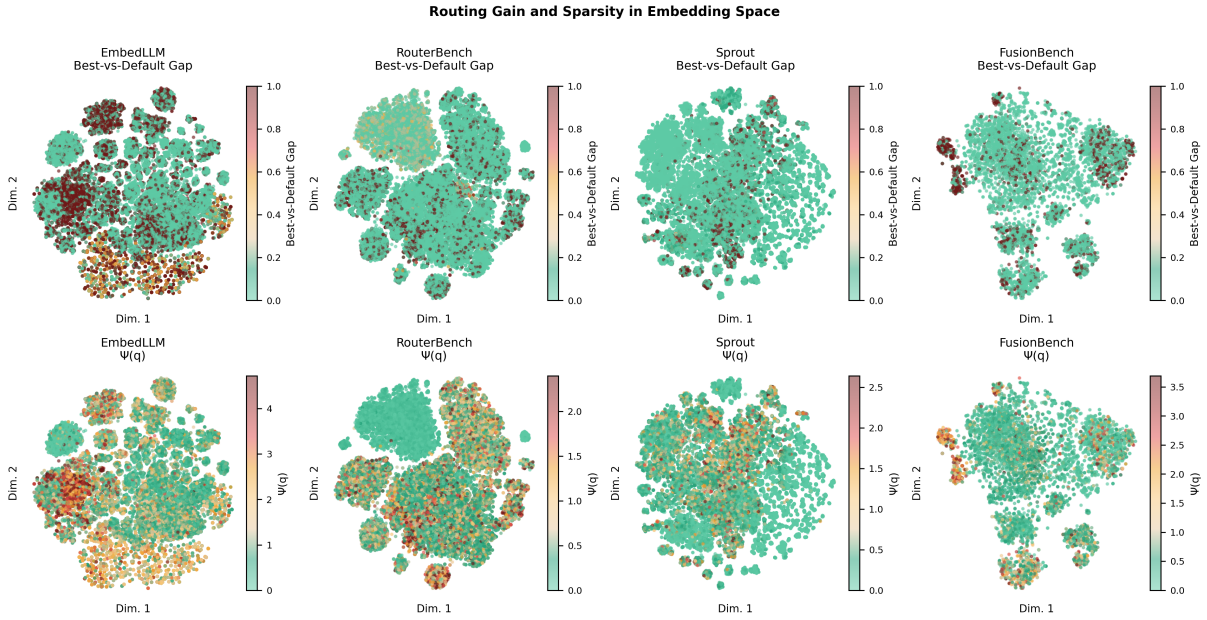


Figure 12: **Performance sparsity $\Psi(q)$ spatially co-localizes with Best-vs-Default-Gap across all benchmarks, validating it as a geometrically predictable acquisition signal.** Each column is one benchmark; each row is a t-SNE projection of query embeddings. *Top*: Best-vs-Default-Gap - score of the per-query best model minus the score of the globally best model; warm colors mark queries where routing strictly improves over the default. *Bottom*: Performance-vector sparsity $\Psi(q) = D_{\mathrm{KL}}(\tilde{\mathbf{s}}(q) \| \mathcal{U})$; high-$\Psi$ clusters overlap with high-gap regions (top row), confirming that Ψ targets exactly the exception regions where routing gains are largest.

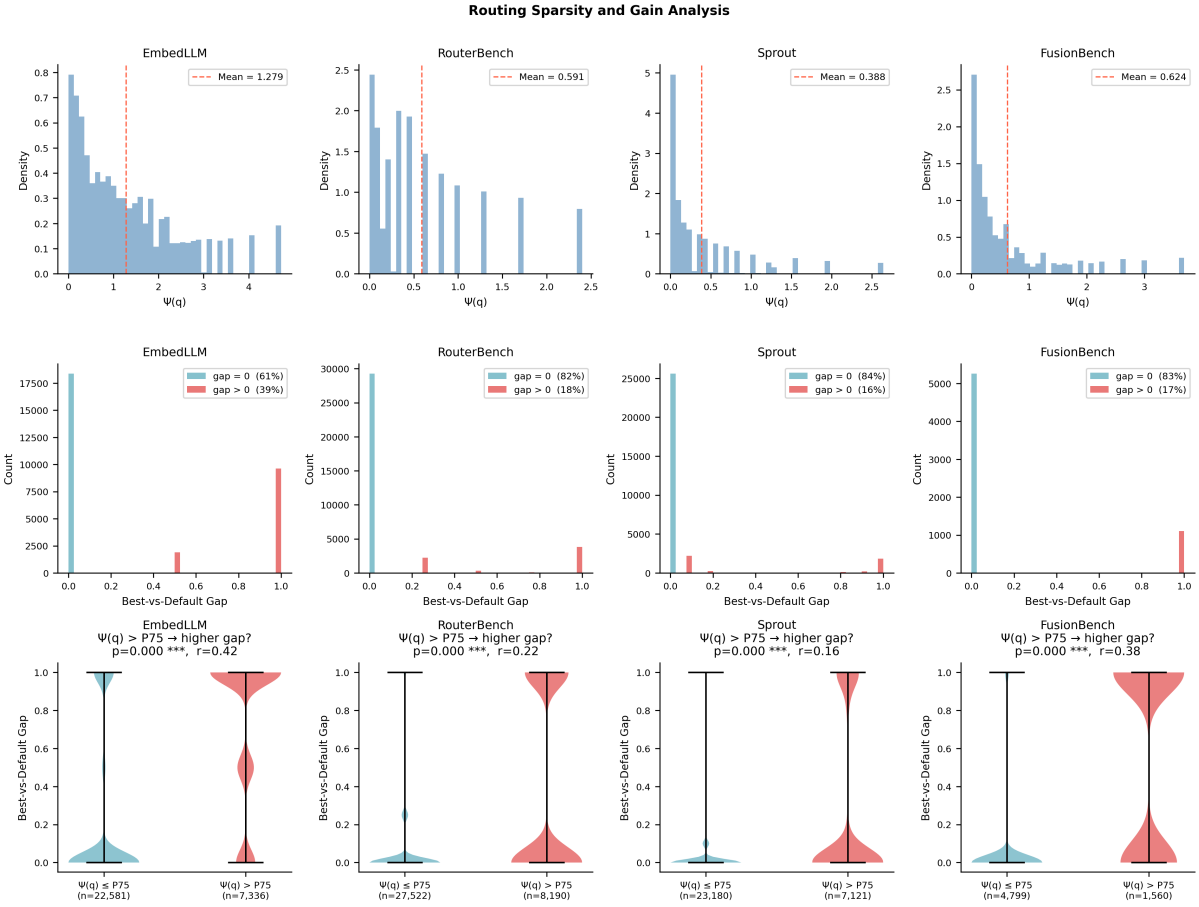


Figure 13: **High-Ψ queries concentrate Best-vs-Default-Gap: the top quartile of sparsity captures the majority of queries where a specialist strictly outperforms the default, on all four benchmarks.** Each column is one benchmark. *Top*: Distribution of $\Psi(q)$; the dashed line marks the mean. The distribution is right-skewed with a heavy tail. *Middle*: Best-vs-Default-Gap distribution; bars indicates queries where routing cannot improve over the default (gap $= 0$), red bars those where it can (gap > 0). *Bottom*: Violin plots of the Best-vs-Default-Gap for low-Ψ ($\leq$ P75) vs. high-Ψ ($>$ P75) queries; Mann-Whitney U p-values ($p < 10^{-3}$) and effect sizes $r \in [0.16, 0.42]$ confirm that high-Ψ queries carry significantly larger routing gains on all benchmarks.