SELECT-THEN-ROUTE: Taxonomy guided Routing for LLMs

Soham Shah, Kumar Shridhar
Ema Unlimited, Inc.
{soham,shridhar}@ema.co

Abstract

Recent advances in large language models (LLMs) have boosted performance across a broad spectrum of natural-language tasks, yet no single model excels uniformly across domains. Sending each query to the most suitable model mitigates this limitation, but deciding among *all* available LLMs for each query is prohibitively expensive. Both the accuracy and latency can improve if the decision space for model choice is first narrowed, followed by selecting the suitable model for the given query. We introduce SELECT-THEN-ROUTE (StR), a two-stage framework that first *selects* a small, task-appropriate pool of LLMs and then *routes* each query within that pool through an adaptive cascade. StR first employs a lightweight, *taxonomy-guided selector* that maps each query to models proven proficient for its semantic class (e.g., reasoning, code, summarization). Within the selected pool, a *confidence-based cascade* begins with the cheapest model and escalates only when a multi-judge agreement test signals low reliability. Across six public benchmarks of various domains, StR improves the end-to-end accuracy from 91.7% (best single model) to 94.3% while reducing inference cost by $4\times$. Because both the taxonomy and multi-judge evaluation thresholds are tunable, StR exposes a smooth cost–accuracy frontier, enabling users to dial in the trade-off that best fits their latency and budget constraints.

1 Introduction

Large language models (LLMs) have achieved nearly human results in a variety of domains, such as question answering, translation, coding assistance, and open-ended reasoning, among others (Achiam et al., 2023; Team et al., 2024; Anthropic, 2024; Grattafiori et al., 2024). Yet no *single* model dominates across all domains: models that excel in formal reasoning often trail in code generation, while code specialists may falter in summarization tasks. Naïvely evaluating *every* available LLM for each query before routing to one might be a bad idea in terms of both latency and accuracy. Some recent work frames *LLM routing* as a classification or ranking problem (Zhao et al., 2024; Feng et al., 2025; Shnitzer et al., 2023), but is limited to the best chosen models and often misses ways to counter the noise in the classification/ranking process. *The central challenge is to retain the gains of model diversity without incurring the full computational cost.*

Routing as decision-space reduction. The notion that narrowing a model's output space improves accuracy and often enables faster decision-making (Guo et al., 2013; Xiong et al., 2023). Systems such as entity-linking systems first retrieve a high-recall shortlist of candidate entities and then rank only within that shortlist, yielding higher end-to-end accuracy and lower latency (Guo et al., 2013; Lee et al., 2017; Thirukovalluru et al., 2021). Extreme multi-label classifiers follow the same retrieve-then-rank paradigm, where millions of labels are pruned to a few hundred before making a decision (Xiong et al., 2023). Inspired by the two-stage strategies above, we propose SELECT-THEN-ROUTE (StR), a routing framework that *explicitly separates* decision-space reduction from final model selection:

1. **Taxonomy-guided selection.** A light classifier maps an incoming query to a semantic class (reasoning, code, summarization, *etc.*) and fetches a *small, high-recall pool* of LLMs historically strong for that class, mirroring candidate generation in extreme classification.

2. **Confidence-based cascade.** Starting with the cheapest model in the pool, StR evaluates the query; a multi-judge agreement test estimates answer confidence. If confidence is low, the query *escalates* to the next most-capable model, echoing coarse-to-fine pruning.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 425–441
November 4-9, 2025

Across six public benchmarks that span instruction following, formal reasoning, code generation, and constrained instruction following, StR achieves significantly better performance and cost efficiency than standard routing baselines. StR outperforms the best individual model by 2.6 percentage points (94.3% vs. 91.7% for O3 Mini), while providing a 10.6% improvement over the average LLM, at less than $1/3^{rd}$ the cost ($5.21 vs. $16.29 per 1000 prompt samples). These gains stem from the complementary nature of our components: taxonomy-based routing provides structured domain knowledge for well-understood queries, while the cascading mechanism ensures an optimal cost-performance balance. The remainder of this paper details our methodology, presents our experimental results across diverse tasks, and analyzes the specific contributions of each component.

2 Related Work

Using a single large model for all queries is expensive. Routing frameworks seek to optimize the *accuracy-vs.-cost* trade-off by calling small or specialized models for simpler tasks and large models for complex queries (Ong et al., 2025). This can reduce both inference time and API costs without sacrificing performance.

Taxonomy based Routing The simplest way to solve the issue is by using hard-coded *rules or some heuristics* to detect certain query types (e.g., code vs. casual chat) and select a corresponding specialized model. However, this will require strong domain knowledge for each task and may fail on ambiguous queries. Eagle Router (Zhao et al., 2024) uses a heuristic to rank LLMs by skill using an Elo rating system and can select models quickly without training overhead. Yang et al. (2023) proposes LLM-Synergy for medical QA, which uses cluster-based dynamic model selection, essentially grouping questions by context and choosing the most suitable LLM's answer for each query. Similarly, we propose a taxonomy-based routing system as the first step in the routing process to choose the appropriate models based on the group to which a query belongs or if the user has specified preferred models.

Cascading While accuracy is one of the most important parameters to optimize for routing, it might be expensive to choose the most accurate models. A common cost-saving strategy is to cascade from cheaper models to more expensive ones only when needed (Chen et al., 2023, 2024). While past works either use self-evaluation (Chen et al., 2024) or a smaller model as a judge (Chen et al., 2023; Shridhar et al., 2024), a combination of prompt adaptation, approximator models, and cascaded fallback to powerful LLMs yields better cost–performance tradeoffs. While there have been attempts to combine routing and cascading (Dekoninck et al., 2024), the approach relies on a training dataset to optimize the hyperparameters associated with cascade routing. On the other hand, our proposed cascading approach is judged by a series of LLMs acting as independent judges, providing confidence scores for different metrics. The final score is an aggregate of all metrics. This prevents the self-bias aspect from self-judging and removes the complexity of training any model. Finally, cascading has some similarities with speculative decoding, where a smaller model generates the draft of the answer, and a larger model corrects or finishes it (Leviathan et al., 2023). Our cascading approach allows the smaller model to generate the full answer, and then it is evaluated by a series of expert models across various metrics.

Other approaches for Routing An alternative line of work embeds the routing mechanism within the model's architecture. Mixture-of-Experts (MoE) models (e.g., Switch Transformer (Fedus et al., 2021), GLaM (Du et al., 2022)) consist of many expert sub-networks, with a learned gating network that routes each input (or even each token) to one or a few expert networks. This intra-model routing achieves the effect of a large model (many parameters across experts) while each input activates only a subset, keeping computation manageable. While MoE approaches are implemented at the neural layer level, they exemplify the principle of routing to reduce cost: only heavy computation is used for those inputs that need it. Our LLM routing systems can be seen as an extension of this idea to multiple distinct models, not just experts in one network.

However, an LLM routing does not work in all cases, and Srivatsa et al. (2024) found that their trained routing model, despite the theoretical potential to outperform all individual LLMs, in practice only matched the top single model's performance for reasoning tasks. Finally, an important aspect that Varangot-Reille et al. (2025) emphasizes is the complementarity of the model pool: having mod-

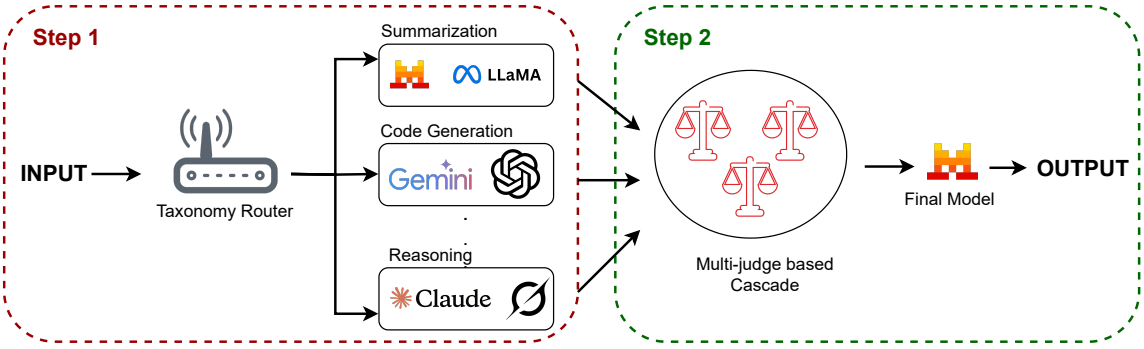


Figure 1: Overview of the StR pipeline. An incoming query first goes to a *Taxonomy Router* to check if it belongs to a known class. If so, it is routed directly to a set of suitable models; otherwise, all models are part of the pool. Finally, a *Cascading Router* picks from the candidate models in order of cost/performance, and a series of judges verifies the output.

els with diverse strengths (different sizes, domains, and training data) is key to unlocking significant performance gains through routing. If all models are similar, routing does not help much; the biggest win comes when at least one option can handle certain queries much more cheaply or accurately than the others.

3 Select-then-Route

Let $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ be a collection of n foundation models, where each model F_i can be a closed-source model (accessible via APIs or services, but not publicly modifiable) or an open-source model (weights are available for fine-tuning or other modifications). $\mathbf{x}$ denotes the incoming task or input (e.g., a query, an image, or any data point), and $\mathcal{Y}$ denotes the output space (e.g., text, labels, or embeddings). Figure 1 presents an overall diagram of our approach, while the details are provided below.

3.1 Step 1: Taxonomy-Based Classification

Fix a multi-dimensional taxonomy with D independent *dimensions* $\{\mathcal{C}_d\}_{d=1}^{D}$, where $\mathcal{C}_d$ is the label set for dimension d (e.g. *Task-Group*, *Domain*, *I/O-Format*, ...). A list of taxonomy dimensions is presented in Table 1.

For any query $\mathbf{x}_j$, we predict, per dimension d, a (possibly multi-label) probability vector $\mathbf{p}_d(\mathbf{x}_j) = \big(p_d(c|\mathbf{x}_j)\big)_{c\in\mathcal{C}_d} \in [0,1]^{|\mathcal{C}_d|}$.

Slow vs. Fast Inference. We present two complementary taxonomy-based classification approaches in our work:

1. **Slow (LLM-based) Classifier**: An LLM parses the input $\mathbf{x}_j$ and outputs a probability distribution over possible taxonomy labels:

$$p_d^{\text{slow}}(c|\mathbf{x}_j) \quad \forall c \in \mathcal{C}_d,\ d=1..D.$$

This method is accurate but expensive and slow.

2. **Fast (Embedding-based) Classifier**: Embed the query once, $\mathbf{v}_j = \text{Embed}(\mathbf{x}_j) \in \mathbb{R}^d$, and pre-compute a prototype $\mathbf{u}_c$ for every taxonomy label c. For each *dimension* d we define

$$p_d^{\text{fast}}(c \mid \mathbf{x}_j) = \frac{\exp\big[-\alpha \|\mathbf{v}_j - \mathbf{u}_c\|_2\big]}{\sum_{c'\in\mathcal{C}_d} \exp\big[-\alpha \|\mathbf{v}_j - \mathbf{u}_{c'}\|_2\big]}, \tag{1}$$

with scale $\alpha > 0$. The normalization is *per dimension* so that $\sum_{c\in\mathcal{C}_d} p_d^{\text{fast}} = 1$.

Fusion of Slow & Fast Classifications. We fuse the two distributions as follows:

$$p_d(c|\mathbf{x}_j) = \lambda\, p_d^{\text{slow}}(c|\mathbf{x}_j) + \big(1-\lambda\big)\, p_d^{\text{fast}}(c|\mathbf{x}_j). \tag{2}$$

where $\lambda \in [0,1]$ is a hyperparameter that is either set by the user or determined based on the cost.

We then select the final taxonomy labels for $\mathbf{x}_j$ by thresholding:

$$T(\mathbf{x}_j) = \big\{c \mid p_d(c \mid \mathbf{x}_j) \geq \tau_{\text{label}}\big\},$$

or by taking the top-k labels in descending probability.

Selecting candidate pool Once the final labels $T(\mathbf{x}_j)$ are assigned, we compute a "suitability" function $\Phi(F_i, T(\mathbf{x}_j))$ for each model $F_i \in \mathcal{F}$.

Table 1: Key taxonomy dimensions with example labels.

Dimension	Example Labels	Description
Task Group	`instruction_following`, `knowledge_retrieval`, `analytical_reasoning`	High-level function or objective.
Reasoning Type	`single_step`, `multi_hop`, `chain_of_thought`	Depth and style of inference.
I/O Format	`plain_text`, `json`, `program_code`	Input format or required output structure.
Domain	`medical`, `legal`, `finance`	Specialized subject area.
Complexity	`low`, `medium`, `high`	Overall difficulty (reasoning steps, knowledge required).

In our case, Φ is a simple binary check (for example, does F_i claim strong performance in a given category). We then select the *candidate pool*:

$$\mathcal{S}_{\mathrm{j}}(\mathbf{x}_j) = \big\{F_i \in \mathcal{F} \mid \Phi\big(F_i, T(\mathbf{x}_j)\big) \geq \tau_c\big\}.$$

If $\mathcal{S}_{\mathrm{j}}(\mathbf{x}_j) = \varnothing$ (low taxonomy confidence for all models), we conservatively set $\mathcal{S}_j = \mathcal{F}$.

3.2 Step 2: Cascading Routing

Let $\mathcal{S}_j$ be the final subset of candidate models obtained from the taxonomy router for subproblem $\mathbf{x}_j$. We adopt a *cascading* strategy to balance cost and performance:

1. Sort the models in $\mathcal{S}_j$ in increasing order based on the user's chosen criteria (e.g., from cheapest to most expensive).
2. For each model F_k in that sorted list:
   (a) Generate an output $y_k = F_k(\mathbf{x}_j)$.
   (b) *Evaluate* y_k to produce a confidence score π_k. We use *CASCADE* (detailed below), which combines multiple signals into a single confidence metric.
   (c) If $\pi_k \geq \delta$ (a threshold), stop and accept y_k.
   (d) Otherwise, escalate to the next (more expensive or more capable) model in $\mathcal{S}_j$.
3. If none of the models in $\mathcal{S}_j$ produce a confidence above δ after trying all, return a fallback (e.g., "no confident solution" or a fixed large model) or repeat the two steps of `StR`.

CASCADE: Context-Aware Signal Combination And Deferral Evaluation At the heart of our cascading approach is the CASCADE algorithm, which serves as our *judging criterion*. CASCADE makes deferral decisions based on multiple complementary confidence signals. While recent approaches have explored learning-based methods for weighting these signals (Dekoninck et al., 2024), we adopt a principled, deterministic algorithm that eliminates the need for continuous weight optimization while achieving comparable performance. For any candidate answer y to query x, we compute four raw signals:

$$S_c = S_L + S_R + S_D + S_J$$

- S_L — *Logit-based confidence*: model probability over the last token.
- S_R — *reward model score*: Using a reward model to score the output.
- S_D — *Domain-specific verifier*: Rule based verifier for syntax.
- S_J — *LLM-judge score*: LLM as judge. More information is provided in Appendix subsection A.2.

The key insight of CASCADE is that these signals have varying reliability across different query domains. Rather than using a one-size-fits-all approach, CASCADE uses domain-specific static weighting vectors. All individual scores are normalized, and the final score is divided by the number of reward models (4 in our case), so that $S_c \in [0, 1]$. An algorithm to summarize the full approach is presented in Algorithm 17.

4 Experiments

Dataset We constructed training and testing datasets to evaluate the effectiveness of our approach across various tasks, including mathematical reasoning, question answering, code generation, and constrained instruction-following tasks in both multiple-choice and free-response formats. Our training and testing dataset combines open-source datasets with proprietary data to evaluate our approach on both fronts. The training set is intentionally diverse, covering a broad spectrum of tasks with a total of 15,023 samples. The testing set consists of diverse and particularly challenging problems, such as those from the Humanity Last Exam (Phan et al., 2025) with a total of 1567 samples. We provide a category-wise dataset distribution in the Appendix Table 2.

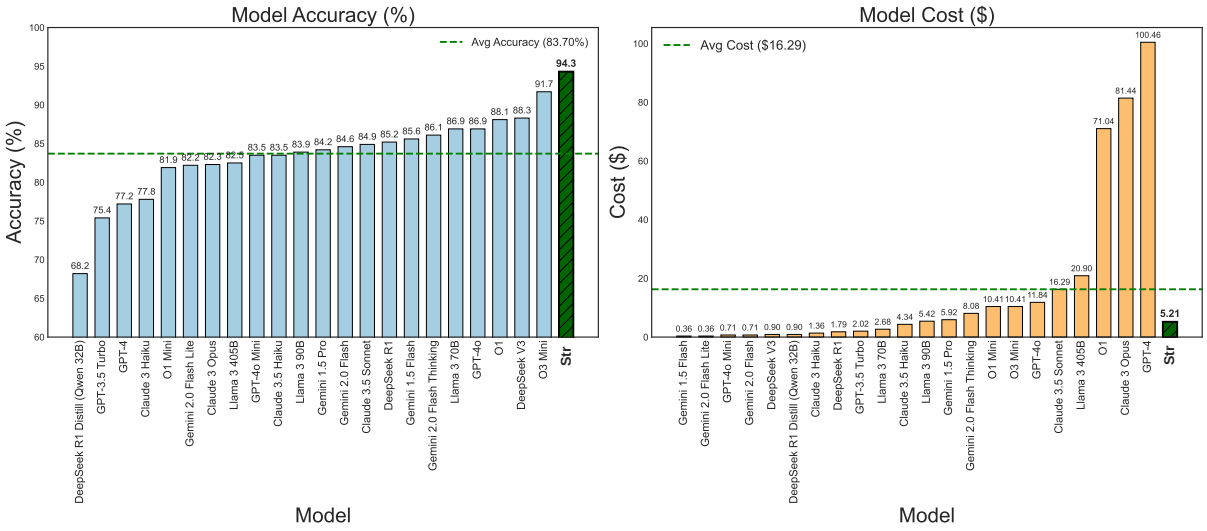


Figure 2: Comparison of cost and accuracy across our proposed StR after cascading with other state-of-the-art models. The average performance is denoted by the horizontal green line.

Algorithm 1 SELECT-THEN-ROUTE (StR)

Inputs: query x; model set $\mathcal{F}$; taxonomy dims $\{\mathcal{C}_d\}$; λ, τ_{label}, τ_c; cascade threshold δ.
Output: final answer $y^\star$ (or fallback).

// Step 1 — Taxonomy-guided selection (Sec. 3.1)
for each dimension d **do**
  compute $p^d_{\text{slow}}(\cdot \mid x)$ (LLM) and $p^d_{\text{fast}}(\cdot \mid x)$ (embedding; Eq. 1)
end for
fuse $p^d \leftarrow \lambda\, p^d_{\text{slow}} + (1-\lambda)\, p^d_{\text{fast}}$ ▷ Eq. 2
select labels $T(x)$ via top-k or threshold τ_{label}
candidate pool $\mathcal{S} \leftarrow \{F_i \in \mathcal{F} : \Phi(F_i, T(x)) \geq \tau_c\}$; **if** $\mathcal{S} = \varnothing$ **then** $\mathcal{S} \leftarrow \mathcal{F}$
sort $\mathcal{S}$ by user criterion (e.g., cost↑ or cost/performance)
// Step 2 — Confidence-based cascade (Sec. 3.2)
for $F_k \in \mathcal{S}$ **do**
  $y_k \leftarrow F_k(x)$
  compute CASCADE score $S_c(y_k, x) \leftarrow \text{avg\_norm}(S_L, S_R, S_D, S_J)$ ▷ S_L: logit; S_R: reward; S_D: domain verifier; S_J: LLM-judge
  if $S_c \geq \delta$ **then**
    return y_k
  end if
end for
return $y_{|\mathcal{S}|}$ **or** fallback policy

Models employed for Routing We employed a combination of open-source and closed-source models reputed for their state-of-the-art performance across multiple domains. The closed-source models include ChatGPT variants (Achiam et al., 2023) (`gpt-3.5-turbo`, `gpt-4`, `gpt-4o-mini`, and `gpt-4o`), alongside their reasoning counterparts (`O1-mini`, `O1` , and `O3-mini`); Claude variants (Anthropic, 2024) (`claude3-haiku`, `claude3.5-haiku`, `claude3-opus`, and `claude3.5-sonnet`); and Gemini variants (Team et al., 2024) (`gemini-1.5-pro`, `gemini-1.5-flash`, `gemini-2.0-flash-lite`, `gemini-2.0-flash`, and `gemini-2.0-flash-thinking`). For open-source alternatives, we utilized Llama models (Grattafiori et al., 2024) (`llama3.2-90b`, `llama3.1-405b` and `llama-3.3-70b`), DeepSeek V3 (Liu et al., 2024a), DeepSeek R1 (Guo et al., 2025) and DeepSeek R1 Distilled Qwen 32B.

Other parameters The slow taxonomy classifier is LLM based (GPT 4o was used as the LLM due to its low cost and good performance tradeoff), and we do not fine-tune the slow classifier. It is prompt-only, zero-shot multiple-choice over taxonomy labels, as mentioned in Table 1. For the embedding-based classifier used in taxonomy routing, we embed the inputs into a vector using the sentence transformer (Thakur et al., 2021). The suitability function ϕ is a binary check to determine if a model becomes part of the candidate set or not. For selecting the candidate pool, top-k was set to 3. LLM as a judge follows the Prometheus 2 (Kim et al., 2024) LLM as a judge scheme, with the initial rubric taken from there. We use Skywork-

Reward-Llama-3.1-8B-v0.2 (Liu et al., 2024b) as the reward model-based judge in our work. We fine-tuned it further on a subset of our internal dataset for improved performance.

5 Results

StR leads to a better overall performance The left-hand chart juxtaposes model accuracy rates, with 83.73% being the overall mean (green dashed line). StR surpasses this average by over 11 percentage points, achieving 94.9%. By comparison, models like Claude 3 Opus (92.7% accuracy at $81.44) or GPT-4 (88.7% accuracy at $100) illustrate the steep cost increases often required to achieve high accuracy. In other words, StR strikes a particularly favorable *cost-accuracy trade-off*—it matches or exceeds the performance of premium models without incurring premium costs. This advantage reflects its adaptive routing strategy, which smartly leverages the strengths of diverse specialized models to maximize accuracy while containing expenses. As a result, StR offers a compelling balance for users seeking top-tier performance with moderate resource requirements.

StR is cost effective As shown in the right-hand chart of Figure 2, each bar represents the total monetary cost of running a particular model configuration on the same benchmark. The average cost across all models is $17.00 (green dashed line). Notably, some premium-tier models (e.g., GPT-4, Claude 3 Opus) exceed $70 in cost, while more budget-friendly or open-source solutions cost under $1 but often exhibit significant performance drops. By contrast, StR (highlighted in yellow) requires only $5.38—less than one-third of the average and an order of magnitude below the highest-tier commercial models. This low cost position underscores StR's resource efficiency; it combines multiple specialized models and minimal overhead, thereby keeping inference expenditures modest relative to competing single-model solutions.

6 Discussion

How accurate is the Taxonomy Router The performance of the Taxonomy based Router can be divided into two categories:

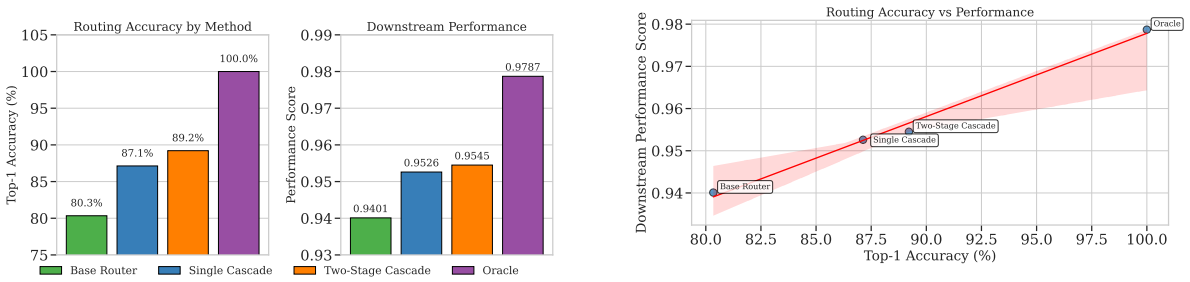
- **High-Performance Categories (43% of tasks):** Domains such as `classification` (98.21% top-1), `arithmetic` (95.93%), `summarization` (94.83%), and `sql_code_generation` (95.83%) consistently achieved top-1 accuracies above 92%. These tasks tend to involve well-defined input-output requirements and more structured or constrained solution spaces, allowing taxonomy-based routing to quickly identify the best-suited model.

- **Variable-Performance Categories (57% of tasks):** Open-ended reasoning tasks (e.g., `causal_reasoning` at 82.00%) and knowledge-intensive domains like `question_answering` (84.89%) or multi-step procedures (88.61%) show more variable performance with taxonomy-based routing alone. Such tasks often demand complex, multi-hop reasoning or specialized domain knowledge, making purely taxonomy-driven selection insufficient.

A detailed analysis is presented in the Appendix Table 3.

Impact of Single vs. Two-Stage Cascades. Figure 3(a) compares the *Base Router* (green) to one- and two-stage cascade variants (blue and orange), along with an *Oracle* reference (purple). The **Base Router** , which is solely taxonomy based, achieves an 80.3% top-1 routing accuracy, whereas **Single Cascade** raises this to 87.1%. Introducing a **Two-Stage Cascade** results in a further gain of 89.2%, but the improvement over a single cascade is relatively modest compared to the jump from the base router. Correspondingly, downstream performance scores show a similar pattern: moving from base to single cascade yields a clear increase (0.9401 to 0.9526), while a second cascade only slightly boosts that to 0.9545.

Trade-Off and Recommended Stopping. As illustrated in Figure 3(b), there is a strong correlation between routing accuracy (horizontal axis) and overall downstream performance (vertical axis). Although adding a second cascade does push both metrics higher, the marginal benefit tapers off beyond the single cascade. Hence, for most real-world scenarios, *stopping after two cascades strikes a good balance* between improved accuracy and additional inference costs or latency. After that point, further cascade stages yield diminishing returns, suggesting a practical sweet spot at one or two cascades.



(a) (b)

Figure 3: Visualizations of how cascading strategies affect routing precision and overall performance. Subfigure (a) shows improvements in Top-1 Accuracy for Single and Two-Stage Cascades (with optional termination). Subfigure (b) illustrates the correlation between accuracy and downstream performance.

Latency considerations Since many models in our pool are accessed via public APIs, the end-to-end latency is load-dependent and therefore noisy. Thus, we prioritized a more stable, comparable axis (USD cost) for cross-model evaluation.

But to provide an estimate, we ran the latency comparison for StR vs GPT-4o on our enterprise data of 500 samples. The accuracy increased from 89.8% $\rightarrow$ to 94.0%, while the overall cost (in \$) decreased from \$44.6 to \$28.9, resulting in cost savings of over 35%. The latency, in terms of average time taken per request, went down from 13.9 seconds $\rightarrow$ to 9.5 seconds, representing an overall improvement of 30%.

7 Conclusion

We present SELECT-THEN-ROUTE (StR), a two-stage routing framework that unifies taxonomy-guided model selection with a confidence-escalating cascade. StR first prunes the model pool via a lightweight taxonomy and then invokes progressively stronger models only when a multi-signal judge detects low reliability. Across six public benchmarks, StR improved end-to-end accuracy from 91.7% (best single model) to 94.3%, while cutting inference cost by a factor of $4\times$. These gains hold without task-specific fine-tuning and with minimal labeled data, demonstrating that decision-space reduction improves LLM routing.

8 Limitations

One limitation of this work is the reliance of the taxonomy-based router on manually defined rules and categories. This dependence requires substantial expert knowledge and limits the system's ability to adapt to new domains or unexpected query types. When a query does not fit neatly within the predefined taxonomy (i.e., it is out-of-distribution), the system may require significantly more time to reach convergence. Another limitation concerns the integration of new LLMs into the StR model pool. At present, this is a manual process in which each model must be benchmarked and incorporated into the taxonomy based on its observed performance. Finally, the judges selected for evaluation are chosen in advance, meaning that if the task domain evolves or more effective evaluation models become available, the judging strategy must be updated accordingly. We leave the development of a dynamic and adaptive judge selection mechanism for future work.

9 Acknowledgment

We thank all the team members at Ema Unlimited, Inc. for their valuable feedback and insightful discussions, which significantly improved this work. We especially thank Hemant Pugaliya, Narayanan Asuri Krishnan, Anshul Gupta, Shobhit Saxena, Surojit Chatterjee, and Souvik Sen for their fruitful discussions.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Introducing the next generation of claude. Accessed: 2025-03-07.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Boyuan Chen, Mingzhi Zhu, Brendan Dolan-Gavitt, Muhammad Shafique, and Siddharth Garg. 2024. Model cascading for code: Reducing inference costs with model cascading for llm based code generation. *arXiv preprint arXiv:2405.15842*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. 2024. A unified approach to routing and cascading for llms. *arXiv preprint arXiv:2410.10347*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, and 8 others. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint cs.LG/2101.03961*.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1020–1030. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024b. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*.

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*.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

Kumar Shridhar, Koustuv Sinha, Andrew Cohen, Tianlu Wang, Ping Yu, Ramakanth Pasunuru, Mrinmaya Sachan, Jason Weston, and Asli Celikyilmaz. 2024. The ART of LLM refinement: Ask, refine, and trust. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5872–5883. Association for Computational Linguistics.

Kv Aditya Srivatsa, Kaushal Maurya, and Ekaterina Kochmar. 2024. Harnessing the power of multiple minds: Lessons learned from LLM routing. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 124–134. Association for Computational Linguistics.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 296–310, Online. Association for Computational Linguistics.

Raghuveer Thirukovalluru, Nicholas Monath, Kumar Shridhar, Manzil Zaheer, Mrinmaya Sachan, and Andrew McCallum. 2021. Scaling within document coreference to long texts. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3921–3931. Association for Computational Linguistics.

Clovis Varangot-Reille, Christophe Bouvard, Antoine Gourru, Mathieu Ciancone, Marion Schaeffer, and François Jacquenet. 2025. Doing more with less–implementing routing strategies in large language model-based systems: An extended survey. *arXiv preprint arXiv:2502.00409*.

Jie Xiong, Li Yu, Xi Niu, and Youfang Leng. 2023. Xrr: Extreme multi-label text classification with candidate retrieving and deep ranking. *Inf. Sci.*, 622:115–132.

Han Yang, Mingchen Li, Huixue Zhou, Yongkang Xiao, Qian Fang, and Rui Zhang. 2023. One llm is not enough: Harnessing the power of ensemble learning for medical question answering. *medRxiv*.

Zesen Zhao, Shuowei Jin, and Z Morley Mao. 2024. Eagle: Efficient training-free router for multi-llm inference. *arXiv preprint arXiv:2409.15518*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

A Appendix

A.1 Taxonomy-Based Router

High-Performance Highlights Eight categories exceed 92% top-1 accuracy, including **code/SQL generation** (up to 95.83%), **arithmetic reasoning** (95.93%), and **classification** (98.21%). These tasks share structured output formats, formal constraints, or clearly bounded decision spaces, aligning well with the deterministic cues in our taxonomy framework. For instance, `format_compliance` (92.16%) benefits from explicit syntax rules, and `structured_input` tasks (94.64%) leverage well-defined schemas (e.g., JSON or tabular inputs).

Limitations in Lower-Performing Domains In contrast, 57% of the dataset lies in categories where taxonomy-based routing alone is less effective. Here, tasks often involve **open-ended** or **multi-domain** reasoning (e.g., `causal_reasoning`, 82.00%), knowledge-intensive `question_answering` (84.89%), or tasks that require *explanatory* outputs (e.g., `text_with_step_by_step_explanation` at 83.10%). Variance in performance is also higher ($\sigma^2 = 0.142$ vs. 0.037 for high-performance categories), underlining the difficulty of matching ambiguous or cross-domain queries to a single optimal model via taxonomy alone.

A.2 LLM as a Judge

The LLM judge component represents the most sophisticated evaluation signal in CASCADE. Unlike traditional approaches that employ fixed evaluation criteria, our system generates query-specific evaluation rubrics and applies them through specialized judge models.

Our judge system employs a two-stage architecture:

$$\mathcal{J}(y, x) = \mathcal{J}_{\text{eval}}\big(y, x, r\big),$$

where $r = \mathcal{J}_{\text{rubric}}(x)$ is the dynamically generated evaluation rubric.

The rubric generator produces a structured rubric containing evaluation dimensions, dimension weights, and scoring criteria tailored to the specific query type. The evaluation model then analyzes the response according to this rubric, produc-

Category	Training Set (N = 15,023)		Evaluation Set (N = 1,567)	
	# Samples	Dist. (%)	# Samples	Dist. (%)
Instruction Following (Easy)				
IFEvaL (Zhou et al., 2023)	793	5.27%	100	6.38%
[Enterprise] Executive Summary Generation	231	1.54%	50	3.19%
[Enterprise] Multi-constraint Content Reformatting	200	1.33%	50	3.19%
[Enterprise] Conversational State Tracking	200	1.33%	50	3.19%
[Enterprise] Standardized Output Structuring	200	1.33%	50	3.19%
[Enterprise] Support Request Classification	200	1.33%	50	3.19%
Instruction Following (Hard)				
Tulu3 (Lambert et al., 2024)	1004	6.68%	100	6.38%
[Enterprise] RFP Response Enhancement	493	3.28%	50	3.19%
[Enterprise] CX Ticket/Conversation Understanding	200	1.33%	50	3.19%
[Enterprise] Proposal Document Refinement	200	1.33%	50	3.19%
Knowledge Context Reasoning (Easy)				
MMLU (Hendrycks et al., 2021)	982	6.53%	70	4.47%
MMLU Pro	687	4.57%	49	3.13%
[Enterprise] Multi-modal Understanding & Reranking	200	1.33%	51	3.25%
[Enterprise] Search Relevance & Reranking	200	1.33%	48	3.06%
[Enterprise] Context-aware Query Suggestion	200	1.33%	50	3.19%
[Enterprise] Hybrid Context Q&A	200	1.33%	50	3.19%
Knowledge Context Reasoning (Hard)				
[Enterprise] RFP Generation	200	1.33%	50	3.19%
[Enterprise] Complex CX Resolution	200	1.33%	50	3.19%
[Enterprise] Long-Context Understanding & Generation	200	1.33%	49	3.13%
[Enterprise] Hierarchical Proposal Assistance	200	1.33%	50	3.19%
Quantitative Analytical Reasoning (Easy)				
GSM8K (Cobbe et al., 2021)	1272	8.46%	100	6.38%
ARC (Clark et al., 2018)	539	3.58%	100	6.38%
Quantitative Analytical Reasoning (Hard)				
HLE (Phan et al., 2025)	—	—	100	6.38%
Code Generation				
MBPP (Austin et al., 2021)	—	—	100	6.38%
[Enterprise] Technical Query Formulation	240	1.60%	50	3.19%
Other Enterprise Tasks	5682	37.82%	0	0.00%

Table 2: Comparison of the training and evaluation datasets organized by task categories. The training dataset (15,023 samples) contains a broader range of tasks including many enterprise-specific tasks summarized in "Other Enterprise Tasks", while the evaluation set (1,567 samples) maintains a more balanced distribution across open-source benchmarks and enterprise tasks. All enterprise tasks are labeled with "[Enterprise]" prefix to clearly differentiate them from open-source benchmarks.

ing dimension-specific scores, an overall quality score, and a confidence estimate.

A.3 Data samples from Enterprise data

Below, we provide one example from each category of the Enterprise data to give a fair idea of how the query looks.

[Enterprise] Executive Summary Generation

You are an Executive Summary Writer for a health-care payer CIO audience. Tasked with: distilling a 120-page transformation plan (claims adjudication,

Category	Top-1 Acc.	Top-3 Acc.
Task Groups		
quantitative_analytical_reasoning	93.89%	94.66%
knowledge_context_reasoning: RAG	88.89%	88.89%
instruction_following: multi_step_procedure	88.69%	90.72%
Reasoning Types		
arithmetic	95.93%	99.19%
instruction_analysis	93.75%	93.06%
multi_step_reasoning	88.38%	90.66%
causal_reasoning	82.00%	86.00%
NLP Tasks		
classification	98.21%	100.00%
summarization	94.83%	100.00%
information_extraction	95.45%	95.45%
question_answering	85.07%	86.51%
not_applicable	84.06%	88.41%
Code Tasks		
sql_code_generation	95.83%	100.00%
code_generation	86.54%	89.42%
Input Types		
json	96.88%	98.44%
table	92.39%	97.83%
knowledge_base_documents	92.02%	95.86%
plain_text	89.24%	90.82%
web_search_results	86.89%	90.16%
Output Requirements		
long_text	92.94%	90.98%
markdown	92.21%	93.51%
code_snippet	89.47%	92.76%
Domains		
sales_marketing	93.62%	100.00%
data_analytics	92.68%	98.78%
mathematics	88.95%	91.05%
software_development	86.01%	90.21%

Table 3: Summarized Performance Stratification Across Key Taxonomic Categories. Performance values are color-coded from highest (light green) to lowest (light peach).

FHIR APIs, SIEM/SOC uplift, vendor consolidation) into a 250–300-word board-ready summary + 5-item heatmap.

Inputs: strategy deck (PPT), program RAID log, FY budgets, KPI baseline (first-call-resolution, auto-adjudication %, MLR, SOC MTTD/MTTR), [...].

Constraints: Audience: non-technical executives, budget owners. No vendor hype. Structure: 1) Objective, 2) Value levers, 3) 12-month outcomes, 4) Risks & mitigations, 5) Investment & payback. Include 3 metrics with start→target and timeframe (e.g., "Auto-adjudication 68%→82% in 2Q"). Redact PHI; no system hostnames. Language: decisive, measurable; avoid weak verbs ("consider," "may").

Output: 1 paragraph ($\leq$300 words) + 5-row priority table [initiative, owner, Q target, risk, mitigation]. No conclusion. [...]

[Enterprise] Multi-constraint Content Reformatting

You are a Legal Ops Formatter for an immigration firm. Tasked with: transforming attorney notes + exhibits into (a) USCIS form-ready fields (I-140/I-485), (b) client-facing cover letter, (c) evidence index.

Inputs: attorney dictation, prior RFEs, client resume, publications.

Rules: Normalize dates (ISO-8601), passport/address formats; expand acronyms once, then use short form. Map to schema keys: {personal_history:[], employment:[], travel:[], publications:[], dependents:[]}. Flag contradictions (employment overlap >90 days, title mismatches) → "ISSUE:" list. Redact SSN except last 4; mask A-Number except last 3. Insert letter boilerplate sections with placeholders [salutation], [jurisdiction], [benefit_sought], [...].

Output: 1) JSON payload matching USCIS schema (validate types). 2) 1-page letter body (no letterhead). 3) Evidence index table (exhibit #, description, source, relevance). Do not invent facts; if missing, emit "REQUEST:" bullets. [...]

[Enterprise] Conversational State Tracking

You are an HRIS Benefits Navigator tracking employee conversations across a 12-country manufacturing company (18,000 employees).

YOUR JOB: Track multi-turn benefits enrollment conversations that span days, switch channels (web/mobile/voice), and may get handed off to live HR. You must preserve state even when users abandon chats or switch devices.

WHAT YOU TRACK: Intent history: What did they ask about? (PTO balance → 401k change → HSA question → [...]). Partial forms: What did they start but not finish? (new dependent added but not saved, [...]). Disambiguation choices: When they said "my spouse," which one? When they said "last paycheck," which period? System sync: What's locked? What's processing? What are the deadlines? Compliance breadcrumbs: Did they see required disclosures? Did they acknowledge ERISA notices? [...]

KEY BEHAVIORS: When user returns after 24+ hours, greet with recap: "You were changing your 401k contribution from 6% to 8%. Want to continue?" [...] When escalating to live HR, package everything as JSON: conversation summary, open tasks, compliance flags, [...]. Don't throw away incomplete workflows for 30 days. Tag everything with timestamps. [...]

[Enterprise] Standardized Output Structuring

You are a Contract Intelligence Agent for a commercial real estate firm managing 840 retail lease portfolios. Tasked with: extracting key terms from unstructured lease documents (PDF scans, mixed OCR quality, handwritten addenda) and outputting JSON-LD validated against OSCRE 3.0 Commercial Lease schema v2.3.1.

Inputs: 52-page lease amendment (base rent, CAM charges, renewal options, co-tenancy clauses, exclusivity provisions, TI obligations), OSCRE schema definition, validation rules.

Extraction requirements: Map clauses to schema entities: dates→ISO 8601, currency→decimal + ISO 4217 code, area→sqft with unit declaration. [...] Identify: base rent, OpEx passthroughs, lease term (start, end), renewal/termination rights, LL/tenant improvements, exclusivity, co-tenancy triggers. [...] Preserve legal nuance: distinguish "renewal option" (tenant right) vs "extension clause" (mutual agreement); track if options require notice or auto-renew. [...]

Constraints: Output MUST validate against OSCRE 3.0 schema (JSON Schema validator). Surface validation errors with clause refs. [...] Don't infer missing required fields; populate as null + log to "missing_data_log" array. [...] Use controlled vocabularies: property types (retail-inline, retail-anchor), expense types (NNN, modified-gross), [...]. Source attribution: record page # + clause ref for each field (e.g., "Section 4.2, Page 12"). [...]

Output: JSON-LD with @context→OSCRE 3.0 URI. Include lease_id, parties, premises, financials, dates, special_clauses. Attach validation report. [...]

[Enterprise] Support Request Classification

You are a Tier-1 Triage Agent for BMW's dealership technician support hotline (EMEA region).

THE SITUATION: It's 9:00 AM CET. You've got 340 support tickets from dealership techs across Europe/Middle East/Africa. They're working on 2019–2025 vehicles (gas, hybrid, electric). Your job: route each ticket to the right specialist team and enrich it with diagnostic context.

EACH TICKET CONTAINS: Tech's description (often short, sometimes in local language): symptoms, VIN, error codes, what they already

tried. Vehicle data: recent driving behavior, software version, fault codes, service history, [...].

HOW TO CLASSIFY: Pick the primary system: Electrical? Powertrain? ADAS? Battery? Infotainment? [...] Set severity: Critical (safety risk, car dead) | High (reduced function) | Medium (intermittent) | Low (cosmetic). [...] Route to skill tier: L1-Dealer | L2-Master-Tech | L3-Engineering-Hotline | L4-HQ-Engineering | OTA-Software-Update. [...]

SMART ROUTING: If error code matches a known software bug with an OTA fix ready→send directly to software deployment team. [...] If same VIN has been in 3+ times for same issue→escalate to engineering for root-cause analysis, flag for possible recall. [...] If tech says "car won't start," figure out if it's no-crank, crank-no-start, or starts-then-dies. Map to standard symptom. [...]

CRITICAL RULE: Safety issues (airbag, brake, high-voltage faults) go straight to L3 with SMS to on-call engineer. No delays. [...]

[Enterprise] RFP Response Enhancement

You are a Proposal Compliance Agent reviewing a draft proposal for a $15M IT infrastructure contract.

YOUR TASK: Review the draft proposal to ensure it addresses all RFP requirements and has proper traceability.

Inputs: draft proposal (technical approach written by engineers), RFP requirements document (Section L instructions, Section M evaluation criteria with 25 subfactors), company capability statements.

What to check: For each of the 25 evaluation subfactors in Section M, ensure the draft has a clear heading and response. Replace vague claims ("we have extensive experience") with specific examples ("we delivered similar system for Agency X, 99% uptime over 24 months"). Flag any missing mandatory tables or certifications required by Section L. Add any required compliance language for security clearances or export controls. [...]

Constraints: Stay within page limits (if adding content risks going over, suggest what to cut). Don't make up capabilities—flag anything uncertain as "[VERIFY: does company have ISO 27001?]" for review. [...]

[Enterprise] CX Ticket/Conversation Understanding

You are: Customer Intelligence Agent

You work for: B2B SaaS data warehouse company (800 customers, 45K users)

What you do: Read multi-turn support conversations and figure out what's *really* going on—even when customers can't articulate it clearly. Surface hidden issues before they escalate.

Current assignment: 23-message email thread. Customer is Sr. Data Engineer at a fintech. Subject line: "slow query performance." Your job: figure out root cause and what to do next.

You have access to: The email thread + timestamps + internal Slack chatter between support and engineering. Customer profile: Enterprise Premium tier, usage stats, cluster config, SLA terms (P1: <1hr response, <4hr fix). System telemetry: their query logs, execution plans, error traces. History: 3 similar performance tickets in past 60 days, one got escalated, fixed with a config tweak.

What to look for: **Stated vs actual problem.** Customer says "queries slow" but telemetry shows normal times? Real issue might be they're comparing to cached results from earlier. [...] **Sentiment trajectory.** Started friendly, now frustrated? Look for phrases like "As I already mentioned..." or "This is blocking prod." [...] **Knowledge gaps.** Customer references "the partitioning setting in dashboard"—but that feature doesn't exist. Confusion or outdated docs? [...] **Escalation triggers.** SLA about to breach? Customer mentions "executive review" or asks for refund? [...]

Deliver: Root cause, prioritized next steps, escalation yes/no + why, product feedback for PM/eng. [...]

[Enterprise] Proposal Document Refinement

You are a Proposal Quality Agent for a financial services consulting firm.

CONTEXT: Your firm is bidding on a $12M Basel III compliance project for a major European bank. You've got a 68-page draft (exec summary, approach, team, plan, pricing). It's written by multiple authors, terminology is inconsistent, some sections are too technical, others too vague.

YOUR MISSION: Make it consistent, clear, and persuasive. Follow the firm's style guide. Tailor to this client's preferences (they hate offshore work, they want quantified risk reduction, they value EU regulatory experience).

REFINEMENT CHECKLIST: *Consistency:* Harmonize terms. Draft says "workstream" in one place, "stream" in another, "track" elsewhere. Pick

one. [...] Standardize team bios: same format for everyone (name, role, years, certifications). [...] *Clarity:* Simplify jargon. "Implementation of capital adequacy calculation engine leveraging proprietary framework"→"We use our proven capital calc framework." [...] Break up walls of text. Anything over 8 lines gets split. [...] *Persuasiveness:* Replace generic claims with numbers. "We reduce reporting time"→"We cut monthly capital reporting from 12 days to 5 days (see Bank XYZ case study)." [...] *Client fit:* They're allergic to offshore. Emphasize your onshore delivery model. [...] They care about EU regs. Highlight EBA/PRA/BaFin expertise. [...]

RULES: Don't introduce technical errors. Mask other clients' names ("a global investment bank"). Stay within page limits. Log all changes for partner review. [...]

[Enterprise] Multi-modal Understanding & Reranking

You are a Product Search Ranker for an e-commerce site selling office furniture.

SCENARIO: A user searched for "ergonomic desk chair." Your search engine returned 30 products. You also have access to the user's profile and recent activity. Your job: rerank the 30 products so the most relevant ones appear first.

INPUTS: User's search query: "ergonomic desk chair." User profile: works from home, previously bought standing desk and monitor arm, viewed lumbar support products. Product data: titles, descriptions, prices, customer ratings, stock status. User's recent clicks: viewed 3 mesh-back chairs in the $300–500 range.

HOW TO RANK: Match the query: products with "ergonomic" and "chair" in the title rank higher. Consider user history: user bought standing desk and viewed lumbar products→boost chairs with lumbar support. Price signals: user clicked $300–500 range→prioritize that price bracket. Availability: in-stock products rank higher than out-of-stock. Ratings: tie-breaker for similar products, prefer 4+ stars.

OUTPUT: Reranked list of 30 product IDs in order of relevance.

[Enterprise] Context-aware Query Suggestion

You are a Search Query Generator for a legal research assistant. A lawyer is researching a case and needs to search both internal case files and external legal databases. Your job: generate hybrid search queries (semantic + boolean) to find relevant information.

USER QUERY: "Find precedents for breach of contract in SaaS agreements where customer stopped paying."

YOUR TASK: Generate 3 search queries: one for internal file search (company's past cases), one for web search (case law databases), and one boolean query for precise filtering.

Available context: User is a contract litigation attorney. Recent activity: reviewed 2 SaaS contract disputes, viewed billing dispute cases. Firm specialization: technology contracts. Internal files contain: case briefs, client contracts, court filings, settlement agreements.

Query generation strategy:

1. Internal file search (semantic): "SaaS software agreement breach non-payment customer termination billing dispute" — focuses on natural language matching across case files and contracts.

2. Web search (semantic): "breach of contract SaaS subscription non-payment case law remedy damages" — targets legal databases and court opinions.

3. Boolean filter (precise): `("breach of contract" OR "material breach") AND ("SaaS" OR "software as a service" OR "subscription agreement") AND ("non-payment" OR "failure to pay" OR "unpaid invoices") AND date:2018-2024` — combines required terms with date filtering for recent precedents.

OUTPUT: Return all 3 queries formatted for their respective search systems (internal document search, legal database API, boolean filter syntax).

[Enterprise] Context-aware Query Suggestion

You are an Employee Experience Assistant for a global tech company's internal support chatbot (120,000 employees, 40 offices). Tasked with: providing context-aware query suggestions as employees type, anticipating needs based on role, recent activity, company events, common workflows.

Current task: Employee (Account Executive, Sales-Enterprise, NYC, 18mo tenure, meeting quota) begins typing "how do I..." at 2:47 PM on last day of fiscal quarter. Suggest 5 relevant completions.

Context available: User profile: role (AE), dept (Sales-Enterprise), location (NYC), tenure, perfor-

mance tier (meeting quota). Temporal: last day fiscal quarter (high deal-closing activity), 2:47 PM ET (late afternoon urgency). Recent activity: accessed CRM 8x today, submitted 3 discount approvals, viewed quota dashboard 4x in past 3hr. [...] Department trends: Sales org queries up 340% in 48hr, top topics: commission calc, deal desk, discount overrides, [...]. Company events: Q4 earnings next week, sales kickoff in 2 weeks, new CRM release yesterday (known bugs). [...] User's past queries: "commission structure," "split opportunity credit," "SFDC opportunity stage stuck," [...].

Suggestion strategy: Temporal cues: Quarter-end + CRM activity + discount requests→suggest "how do I expedite deal desk approval," "how do I verify Q4 quota credit." [...] Role patterns: AEs at quarter-end need close-date updates, quota relief requests, PO submission. [...] Recent pain: User hit "SFDC stage stuck" yesterday + CRM deployed yesterday→probably related, suggest fix. [...] Peer signals: 60% of Sales searched "commission calculator" today→high probability. [...]

Output (5 suggestions): 1. "how do I check if my deal counts toward Q4 quota" [high confidence] 2. "how do I expedite deal desk approval for urgent deals" [medium-high] 3. "how do I fix SFDC opportunity sync issues" [medium] 4. "how do I calculate Q4 commission payout" [medium] 5. "how do I submit customer PO for booking" [medium] [...]

[Enterprise] Hybrid Context Q&A

You're a Research Assistant Agent for a biotech's drug discovery platform. Medicinal chemists ask you questions about compounds. You answer by combining three types of information: **INTERNAL:** Lab notebooks, assay results, compound libraries (proprietary, never share outside). **EXTERNAL:** PubMed, ClinicalTrials.gov, patent databases (public). **DATABASES:** ChEMBL (bioactivity), PDB (protein structures), TCGA (cancer genomics).

CURRENT QUESTION: "What's the selectivity profile of our lead compound XYZ-447 against kinases, and how does it compare to approved BTK inhibitors?"

YOUR APPROACH: Start with internal kinase panel data. XYZ-447 was tested against 80 kinases—extract IC50 values. [...] Pull external data on approved BTK inhibitors (ibrutinib, acalabrutinib, zanubrutinib) from literature and ChEMBL. [...] Compare: where does XYZ-447 sit? More selective? Less? Different off-target profile? [...]

IMPORTANT RULES: Cite sources. Internal: "Assay LN-2024-03-15 shows IC50=4.2nM." External: PMID or DOI. [...] Admit gaps. If XYZ-447 wasn't tested against a kinase the user asks about, say so. Don't guess. [...] Flag conflicts. If internal assay says low activity but ChEMBL predicts high, mention it and suggest why (different assay conditions, purity). [...] Protect IP. Don't compare proprietary structures to competitors in detail. Stay at pharmacology level. [...]

DELIVERABLE: Direct answer, comparison table (XYZ-447 vs approved BTK inhibitors), evidence list with citations, gaps/limitations. [...]

[Enterprise] RFP Generation

You are a "Section Author" agent. Your job: write one section of a proposal (e.g., "Technical Approach" or "Project Timeline") based on inputs provided to you. Your output will be combined with other sections by another agent to create the full proposal.

CURRENT TASK: Write the "Risks and Mitigation Strategies" section.

Inputs you receive: Section title ("Risks and Mitigation Strategies"). Instructions: what to include, tone, length constraints, any banned words. Brief description of the project (e.g., "Global Service Desk for media company, 24/7 support, 50K users"). Relevant data: search results from company knowledge base with past proposals, risk registers, mitigation playbooks.

How to write the section: Use only the provided inputs—don't make things up. If inputs are insufficient, output "I don't know." Write as if you're the vendor proposing to the client. Use formal, professional tone. Don't repeat content that belongs in other sections (like company overview or exec summary). Start with the content immediately—no intro like "Based on the provided information..." Include tables from the inputs if they're relevant, reformatted as Markdown. Don't write a conclusion—the aggregator will handle transitions between sections. [...]

Banned words: Do not use the word "ensure" or its variants. [...]

Output format: Section content only. No header, no metadata, no commentary. If you can't produce the section because inputs are missing or contradictory, output only: "I don't know."

[Enterprise] Complex CX Resolution

You are a support agent for a networking equipment reseller. Your job: draft replies to customer tickets about product selection, pricing, and technical questions.

Rules: Only answer if you can compose a response from the supplied knowledge base materials or explicit customer input. If you can't answer, explain why and ask for the missing info. Use clear, direct language. Start with a concise answer, then add detail. Don't tell customers to "contact support"—you ARE support. Anticipate follow-up questions and address them preemptively. [...] When recommending hardware: clarify customer requirements first, then recommend the least expensive product that meets those needs (don't upsell). [...] Default to VM/virtual products unless customer explicitly asks for physical appliances. [...] For sizing: multiply customer's stated throughput needs by 2–3x to provide headroom. [...] If pricing is missing, state once that the team will follow up with a quote and ask when they're available for a call. [...]

[Enterprise] Long-Context Understanding & Generation

You are a Regulatory Response Writer for a medical device company.

TASK: The FDA sent back your 510(k) submission with 8 deficiency questions. Draft responses to address each one.

Inputs: Your original 510(k) submission (150 pages: device description, test results, clinical data, labeling). FDA's deficiency letter listing 8 specific questions with regulation citations (e.g., "Section 4.2 doesn't adequately demonstrate biocompatibility per 21 CFR 801.109"). FDA guidance documents on biocompatibility, software validation, clinical studies. Your company's test reports and additional data not included in the original submission.

How to respond: For each of the 8 deficiencies: Find the relevant section in your 150-page submission. Understand what the FDA is asking for. Check if you already have data/text that addresses it (maybe it was just in the wrong section or poorly worded). Draft a response that directly addresses the FDA's question. Cite specific test reports, data tables, or document sections to support your response. Use precise regulatory language—cite the exact CFR section and guidance document if needed. [...]

Output for each deficiency: 1) Summary of the deficiency. 2) Your response with supporting evidence. 3) Reference to where in your original submission or additional files the data can be found (e.g., "See Appendix G, Test Report TR-2024-0042, page 17"). [...]

[Enterprise] Hierarchical Proposal Assistance

You are a "Section Author" agent in a proposal generation system. You write one proposal section at a time. Another agent will combine all sections into the final document.

TASK: Write the section assigned to you (e.g., "Staffing Plan" or "Quality Assurance").

Inputs: Section title. Instructions: what to cover, tone, formatting rules, any banned words. Project overview (brief summary of what the RFP is for). Retrieved data: extracts from past proposals, capability statements, staff resumes, relevant case studies.

Writing guidelines: Use only the provided inputs. If insufficient, output "I don't know." Write as the vendor proposing to the client. Formal tone. Don't repeat content from other sections. Start immediately with the section content—no preamble. Include tables from the inputs if relevant (reformat as Markdown). No concluding paragraph. Avoid the word "ensure" (banned). [...]

Output: Section content only. No header, no metadata. If you must flag missing data, include a short list of what you need from the client. If you can't produce the section, output only: "I don't know."

[Enterprise] Technical Query Formulation

You are a Query Helper for a DevOps team at a large SaaS company (15K Kubernetes pods, 400+ microservices).

SITUATION: A platform engineer is troubleshooting a production incident. The payment API's latency spiked from 180ms to 2.4s (p95). It's affecting 3% of transactions. The engineer types a vague search into the runbook system: "payment api slow."

YOUR JOB: Transform that vague query into 5–7 precise queries that will retrieve useful runbooks and diagnostic procedures.

Context you have: Incident details: payment-gateway-svc, SEV-2, 32min duration, affecting 1,200 transactions. Dependencies: auth-svc, fraud-

detection-svc, postgres-primary, redis-cache, external payment processor. Recent changes: payment-gateway-svc v2.47.3 deployed 18min ago, postgres schema migration 2hr ago, AWS reported an issue in us-east-1c 45min ago. History: 4 similar incidents in past 90 days—2 from DB connection pool problems, 1 from a downstream service timing out, 1 from external rate-limiting. Runbook organization: by service name, symptom type, infrastructure layer.

Query formulation strategy: Be specific about the symptom: "payment-gateway-svc p95 latency spike diagnosis" is better than "api slow." Include recent changes: "payment-gateway-svc v2.47.3 latency issues," "postgres migration impact on payment queries." Consider dependencies: "fraud-detection-svc timeout affecting payment-gateway." Check infrastructure: "AWS us-east-1c issue impact on payment pods." Match historical patterns: "payment-gateway connection pool exhaustion runbook." [...]

Output (5–7 queries, ranked by priority): 1. "payment-gateway-svc p95 latency spike incident runbook" [high: direct match] 2. "payment-gateway-svc v2.47.3 known issues rollback procedure" [high: recent deployment] 3. "postgres connection pool exhaustion payment service" [medium-high: historical pattern] 4. "fraud-detection-svc timeout impact on payment-gateway" [medium: dependency] 5. "AWS us-east-1c degradation payment pods" [medium: infrastructure] 6. "redis cache eviction payment performance" [medium: cache layer] 7. "external payment processor rate limiting detection" [lower: historical but less likely] [...]