OG-Rank: Learning to Rank Fast and Slow with Uncertainty and Reward-Trend Guided Adaptive Exploration

Praphul Singh **Corey Barrett** **Sumana Srivasta**
Irfan Bulu **Amitabh Saikia** **Sri Gadde** **Krishnaram Kenthapadi**
Oracle Health & AI

Abstract

Clinicians need ranking systems that work in real time and still justify their choices. Motivated by the need for a low-latency, decoder-based reranker, we present OG-Rank, a single-decoder approach that pairs a pooled first-token scoring signal with an uncertainty-gated explanation step. The model scores all candidates in one pass and generates a brief, structured rationale only when the list is genuinely ambiguous, keeping latency predictable. Trained with a curriculum that concentrates effort on hard cases, OG-Rank delivers strong effectiveness on encounter-scoped order selection (fast path: Recall@1 0.45, nDCG@20 0.625) and improves further when the gate activates (Recall@1 0.56, nDCG@20 0.699 at a 45% gate rate), while compact backbones show similar gains under the same policy. Encoder baselines trail in both effectiveness and flexibility. The result is a practical recipe: rank fast by default and explain when it helps, a pattern that applies broadly to decision tasks where selective generation buys accuracy at acceptable cost. The single-policy design simplifies deployment and budget planning, and the curriculum principle (spend more on the hard cases, less on the easy ones) readily transfers beyond clinical order selection.

1 Introduction

Clinical order selection is a ranking problem under tight latency and high-stakes trust. Systems must score many candidates fast enough for interactive use while producing justifications clinicians can verify. Classical IR delivers low-latency scores but weak explanations (Robertson and Zaragoza, 2009; Formal et al., 2021; Khattab and Zaharia, 2020; Karpukhin et al., 2020; Devlin et al., 2019; Nogueira and Cho, 2019; Nogueira et al., 2020). Decoder-based LLMs provide rich rationales yet are costly when generated for every candidate (Sun et al., 2023; Ma et al., 2023; Lin et al., 2025). Recent work shows that the first generated token can carry strong ranking signal, but coupling that fast decision with faithful explanation and predictable budgets remains open (Reddy et al., 2024; Chen et al., 2024).

We address speed, uncertainty, and domain safety in one model. First, we keep listwise scoring cheap by reading the *pooled first-token* yes/no signal (log-odds at the first generated position) to rank candidates *without decoding* (Sun et al., 2023; Wei et al., 2022; Wang et al., 2023). Second, we compute a *listwise* uncertainty over the candidate set and route only ambiguous queries to a slow path that returns a structured decision. Third, we align the model to clinical validity, specificity, and safety constraints during training and inference (Bodenreider, 2004; Mandel et al., 2016; Johnson et al., 2016).

We introduce OG-Rank, a single-decoder model initialized from Llama 3.2 3B Instruct (AI at Meta, 2024; Grattafiori and et al., 2024). Candidate sets are built per encounter by subclustering the transcript around each signed order, synthesizing a compact context, extracting a command-like query, and running encounter-scoped embedding search. We keep the top-20 candidates; if the signed order is absent, we retain the top-19 and insert the signed order as the twentieth *during training*. Each set is converted to pointwise instances by selecting one candidate as the target and listing the remainder as context. The instance with the signed order as target is labeled as the positive class, and all others as negative; at generation time we enforce a single decision token where `yes` corresponds to the pointwise target and `no` otherwise. The same retrieval pipeline is used at inference time without oracle insertion.

The model operates at two speeds. The fast path reads pooled first-token yes/no probabilities at the first generated position to produce scores with no

decoding, enabling efficient listwise ranking. The slow path is gated by listwise uncertainty computed over the full candidate set with a fixed threshold $T = 0.9$ used for all backbones, and it generates a single JSON object that contains a total order over candidates and a brief rationale. The JSON ranking is taken as final when present and valid, with a fallback to the fast scores otherwise. This preserves interpretability while keeping serving budgets predictable (Lin et al., 2025; Wei et al., 2022; Wang et al., 2023).

Training uses group-relative policy optimization with a strict multi-axis judge and KL regularization to a frozen pre-epoch checkpoint to align decisions and rationales under domain constraints (Stiennon et al., 2020; Ouyang et al., 2022; Schulman et al., 2017; **?**; Yuan et al., 2023; Ethayarajh et al., 2024; Hong et al., 2024; Liu et al., 2024; Zheng et al., 2024; Humphreys-Read et al., 2024). Adaptive exploration allocates more rollouts and rationale budget where first-token uncertainty or the previous epoch's reward trend indicate difficulty, improving sample efficiency and safety-focused alignment (Miao et al., 2024; Lambert et al., 2024; Wang et al., 2024; Rao et al., 2024; Sener et al., 2024). We situate OG-Rank within clinical NLP by leveraging standard resources and open backbones for transfer (Team et al., 2024; Research, 2024; Lee et al., 2020; Gu et al., 2021).

2 Related Work

Encoder rankers (bi/cross and late interaction). Classical lexical BM25 remains a strong first stage (Robertson and Zaragoza, 2009). Neural rerankers improve effectiveness via cross-encoders (e.g., BERT) and dense retrieval (Devlin et al., 2019; Karpukhin et al., 2020; Nogueira and Cho, 2019; Nogueira et al., 2020), with late-interaction methods (ColBERT) and sparse expansion (SPLADE) offering quality–latency trade-offs (Khattab and Zaharia, 2020; Formal et al., 2021). Benchmarks and tooling have standardized training and evaluation (Nguyen et al., 2016; Thakur et al., 2021; Izacard et al., 2022; Bonifacio et al., 2022; Lin et al., 2021).

Decoder LLM rerankers. Listwise prompting shows strong zero- and few-shot quality but high cost (Sun et al., 2023). FIRST uses single-token decoding to read the first output's logits for listwise scoring, retaining quality while improving speed (Reddy et al., 2024). Independent reproduction confirms speed and quality trends and broadens evaluation (Chen et al., 2024). LLM4Rerank explores graph-structured chain-of-thought for multi-aspect reranking (Gao et al., 2024). Open rerankers from the Qwen3 family provide compact 0.6B–8B options for embedding and reranking (Zhang and the Qwen Team, 2025; Qwen Team, 2025).

Clinical context. Clinical decision support introduces domain-specific constraints related to appropriateness, safety, and specificity, motivating uncertainty-aware gating of explanations and judge-based training. Recent healthcare evaluations highlight the importance of guideline grounding and conservative behaviors in medical workflows, reinforcing the need for structured evaluation under domain constraints (Siepmann et al., 2025; Gaber et al., 2025).

Positioning OG-Rank. OG-Rank combines a pooled first-token score with uncertainty-gated rationales in a single-decoder policy, targeting encoder-like latency for scoring while reserving generation for borderline cases. Candidate sets are formed at the encounter level via LLM-guided subclustering of transcript chunks, synthesis of a compact context, extraction of a command-like query, and encounter-scoped embedding retrieval with oracle inclusion during training. This complements single-token listwise rerankers (Reddy et al., 2024; Chen et al., 2024) and keeps deployment simpler than two-model stacks that separate encoding and explanation.

3 Methods

3.1 Problem Setup and Data Construction

Each encounter provides a transcript as a list of chunks $\mathcal{T} = \{t_k\}$ and a list of signed orders $\mathcal{S} = \{s_\ell\}$. For a particular signed order $s^\star \in \mathcal{S}$ we ask an LLM to select a subcluster $\mathcal{T}^\star \subset \mathcal{T}$ whose content most directly supports $s^\star$. We synthesize a compact context c by concatenating the chunks in $\mathcal{T}^\star$, then extract from c a command-like query q that names the intended action. We run encounter-scoped embedding search with q to obtain the top-20 candidates $\{o_1, \ldots, o_{20}\}$. If $s^\star$ is not in the top-20, we keep the top-19 and include $s^\star$ as the twentieth *during training* to guarantee the presence of the signed order.

We convert each top-20 set to pointwise instances. For each candidate o_j we render a user message

that highlights the target candidate and lists the remaining candidates as compact references:

```
CONTEXT: c
PATIENT: p
CandidateOrder: o_j
OtherCandidates: List(O\{o_j})
```

Here p denotes available patient signals when present, and $\mathrm{List}(\cdot)$ denotes a concise enumeration (e.g., names or short identifiers) truncated for length.

Labels are assigned per pointwise instance: if $o_j = s^\star$ the instance is the *positive class* (target), otherwise *negative*. A decoder-only LM with parameters θ defines next-token distributions. Let $x_{1:T}$ denote the concatenation of the rendered prompt tokens and any generated tokens, and let

$$p_\theta(x_t \mid x_{<t}, u_j) \;=\; \mathrm{softmax}\big(\ell_t(x_{<t}, u_j; \theta)\big)$$

be the distribution over the vocabulary at step t, with logits $\ell_t \in \mathbb{R}^V$. We use left padding for batching and track the first generated position s per sample.

The model must emit a *decision token* $d \in \{\text{yes}, \text{no}\}$ at step s, followed by a textual *rationale* $r = (r_1, \ldots, r_L)$ on the same line. We enforce the convention that `yes` means "the highlighted CandidateOrder is the target/positive" and `no` otherwise. Because tokenization may admit single-token variants for `yes` and `no`, we precompute variant sets $\mathcal{Y}$ and $\mathcal{N}$ (token ids) and pool their probabilities at s.

Pooled first-step probabilities and log-odds. Let $\ell = \ell_s(x_{<s}, u_j; \theta)$ be the first-step logits, and write $\mathrm{softmax}(\ell)_v$ for the probability of vocabulary id v. Define

$$\log p_\theta(\text{yes} \mid u_j) = \log \sum_{i \in \mathcal{Y}} \mathrm{softmax}(\ell)_i, \qquad (1)$$

$$\log p_\theta(\text{no} \mid u_j) = \log \sum_{k \in \mathcal{N}} \mathrm{softmax}(\ell)_k, \qquad (2)$$

and the first-step log-odds

$$z(u_j; \theta) \;=\; \log p_\theta(\text{yes} \mid u_j) \;-\; \log p_\theta(\text{no} \mid u_j). \qquad (3)$$

The fast path does not decode; it reads the first-step probability as

$$s(u_j; \theta) \;=\; \sigma\big(z(u_j; \theta)\big), \qquad (4)$$

where σ is the logistic function, and the associated *per-sample* uncertainty proxy is the Bernoulli variance

$$q(u_j; \theta) \;=\; 4\, s(u_j; \theta)\,\big(1 - s(u_j; \theta)\big) \;\in\; [0, 1]. \qquad (5)$$

3.2 Judge, Reward Scaling, and Hard Vetoes

Given a completion $t = (d, r)$, a strict LLM judge returns rubric-specific boolean vectors $\mathbf{b}_m(t) \in \{0,1\}^{K_m}$ for $m \in \{\text{decision}, \text{clinical}, \text{specificity}, \text{safety}, \text{format}\}$. We convert each vector to a bounded score $S_m(t) \in [-3, 3]$ in two steps.

Weighted success ratio. Let $\mathbf{w}_m \in \mathbb{R}^{K_m}_{\geq 0}$ be non-negative weights. Define

$$r_m(t) \;=\; \frac{\langle \mathbf{w}_m, \mathbf{b}_m(t) \rangle}{\max\{1,\ \|\mathbf{w}_m\|_1\}} \;\in\; [0, 1].$$

Affine map to a symmetric range. Map to $[-3, 3]$ via

$$S_m(t) \;=\; \alpha\, r_m(t) + \gamma, \quad S_m(t) \in [-3, 3],$$

with fixed α, γ chosen so $r_m \in \{0, 1\}$ maps to the endpoints.

Hard vetoes. If a designated veto check fails in a rubric (for example catastrophic safety or malformed first token), we set that rubric's $S_m(t)$ to the lower bound, overriding the affine map.

Composite and decision-only scores. The composite score used for trend analysis and optimization is

$$R(t) \;=\; \sum_m w_m\, S_m(t),$$

with non-negative weights $\{w_m\}$.

3.3 GRPO with KL Regularization

For a fixed prompt u_j we sample n rollouts $t^{(i)}$ with temperature τ and nucleus threshold p, truncated to a rationale budget of L tokens. Let the scalar rewards be $R^{(i)} = R\big(t^{(i)}\big)$ and the group-centered advantages

$$\bar{R} = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} R^{(i)}, \quad A^{(i)} = R^{(i)} - \bar{R}.$$

Let $\mathcal{G}^{(i)}$ index generated token positions of $t^{(i)}$. At the start of each epoch we freeze a reference model

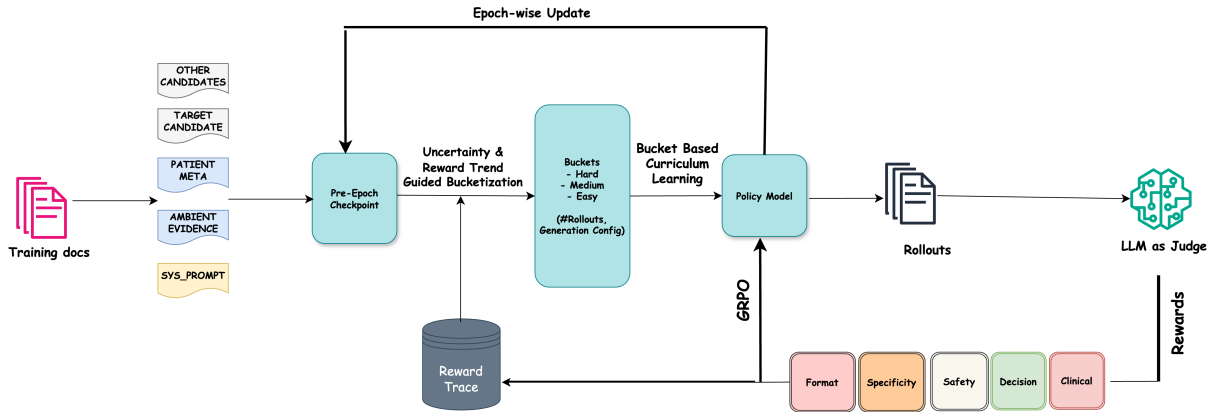

Figure 1: GRPO with uncertainty- and reward-trend-guided curriculum. First-step uncertainty determines the curriculum bucket for the next epoch together with the previous epoch's reward trend. Each bucket uses a fixed rollout configuration; group-relative centering and KL regularization to a pre-epoch reference stabilize updates under a multi-axis clinical judge.

π_{ref} as the pre-epoch checkpoint and keep it fixed during that epoch. The objective per rollout is

$$\mathcal{L}^{(i)}_{\text{grpo}}(\theta) = -A^{(i)} \sum_{t\in\mathcal{G}^{(i)}} \log \pi_\theta\big(y^{(i)}_t \mid u_j, y^{(i)}_{<t}\big) + \beta \sum_{t\in\mathcal{G}^{(i)}} \Big(\log \pi_\theta\big(y^{(i)}_t \mid u_j, y^{(i)}_{<t}\big) - \log \pi_{\text{ref}}\big(y^{(i)}_t \mid u_j, y^{(i)}_{<t}\big)\Big). \tag{6}$$

with $\beta > 0$ controlling KL strength. Group centering reduces variance and the KL term controls drift. *Prompting details.* The rubricized system prompts used for the judge and rollouts are provided verbatim in Appendix B (§B.3).

3.4 Curriculum Learning with Uncertainty and Reward Trend

Motivation. Uniform exploration spends compute on obvious cases and dilutes useful gradients. A curriculum that routes more rollouts and longer rationales to ambiguous or underperforming prompts concentrates learning where the policy is uncertain or failing while keeping obvious cases cheap.

Why bucketization helps. Bucketization partitions prompts into a small number of regimes with fixed decoding budgets. Within an epoch, each bucket uses a fixed rollout configuration $\mathcal{C}_b = (n_b, \tau_b, p_b, L_b)$. Costs are predictable, variance from per-step scheduling is reduced, and difficult prompts are targeted by recomputing buckets between epochs.

Per-sample trend statistic. Let $\mathcal{T}_e(u_j)$ be the multiset of all judged rollouts collected for prompt u_j during epoch e. Define the epoch mean trend

$$\bar{r}_e(u_j) = \frac{1}{|\mathcal{T}_e(u_j)|} \sum_{t\in\mathcal{T}_e(u_j)} \rho(t), \tag{7}$$

where $\rho(\cdot)$ is either $S_{\text{decision}}(\cdot)$ or the composite $R(\cdot)$. We log every tuple $(u_j, t, \{S_m\}, R)$ to a JSONL trace, so $\bar{r}_e(u_j)$ is computed exactly from the log.

Uncertainty statistic. With parameters θ_e at the start of epoch e, compute $s(u_j;\theta_e)$ and $q_e(u_j) = q(u_j;\theta_e)$ for all pointwise prompts.

Bucket assignment rule for epoch $e{+}1$. Choose reward thresholds and uncertainty cutoffs:

$$r_{\text{hard}} < r_{\text{med}} < r_{\text{easy}}, \qquad 0 \le q_{\text{med}} < q_{\text{hard}} \le 1.$$

We assign buckets by combining uncertainty and the previous epoch's trend:

$$b_{e+1}(u_j) = \begin{cases} \text{hard}, & q_e(u_j) \ge q_{\text{hard}} \text{ or } \bar{r}_e(u_j) < r_{\text{hard}}, \\ \text{medium}, & q_{\text{med}} \le q_e(u_j) < q_{\text{hard}} \\ & \text{or } r_{\text{hard}} \le \bar{r}_e(u_j) < r_{\text{med}}, \\ \text{easy}, & \text{otherwise.} \end{cases} \tag{8}$$

Procedural details. A precise, step-by-step version of the curriculum procedure is provided in Appendix A.1 (Algorithm 1).

3.5 Uncertainty-Gated Two-Speed Inference

At test time we form an encounter-scoped candidate list $\mathcal{O}$ by subclustering the transcript, synthesizing a compact context, extracting a command-like query, and running retrieval. The model first computes first-step scores $P_f(j) = \sigma(z(u_j))$ for

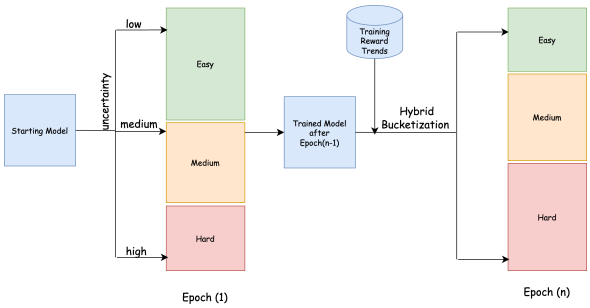

Figure 2: Rebucketing across epochs. At the start of epoch $e{+}1$, each pointwise sample is reassigned to *hard*, *medium*, or *easy* using both uncertainty $q_e(u_j)$ and the previous epoch's mean reward $\bar{r}_e(u_j)$.

each pointwise prompt u_j without decoding. We convert the corresponding log-odds to a listwise distribution $\tilde{\mathbf{p}} = \mathrm{softmax}(\mathbf{z})$ over candidates and measure normalized entropy

$$U \;=\; -\sum_{j=1}^{m} \tilde{p}_j \log \tilde{p}_j \;/\; \log m,$$

which upper-bounds at 1 when candidates are indistinguishable and approaches 0 when one candidate dominates. A fixed uncertainty cap $T = 0.9$ is used across all backbones. If $U \le T$ we return the fast ranking by sorting $\{z(u_j)\}$ in descending order.

If $U > T$ we invoke a one-shot listwise slow path that generates a single JSON object containing a total order (by stable candidate IDs) over all candidates and a brief rationale. When the JSON is valid and covers the full set, the induced order is taken as final. When parsing fails or coverage is incomplete, the system falls back to the fast ranking. We log whether the slow path was triggered per query; `gate_trigger_rate_pct_avg` reports the percentage of queries that crossed the gate, enabling direct comparison of quality–cost trade-offs across models at a common T.

Prompting details. The minimal pointwise fast-path prompt and the listwise JSON slow-path prompt are included verbatim in Appendix B (§B.1 and §B.2).

3.6 Hyperparameters, Implementation, and Metrics

Training. We fine-tune for five epochs with GRPO and a per-epoch frozen KL reference (β=0.02). LoRA on attention/MLP (rank 16, α=32, dropout 0.1). AdamW, lr $1{\times}10^{-6}$, weight decay 0, grad

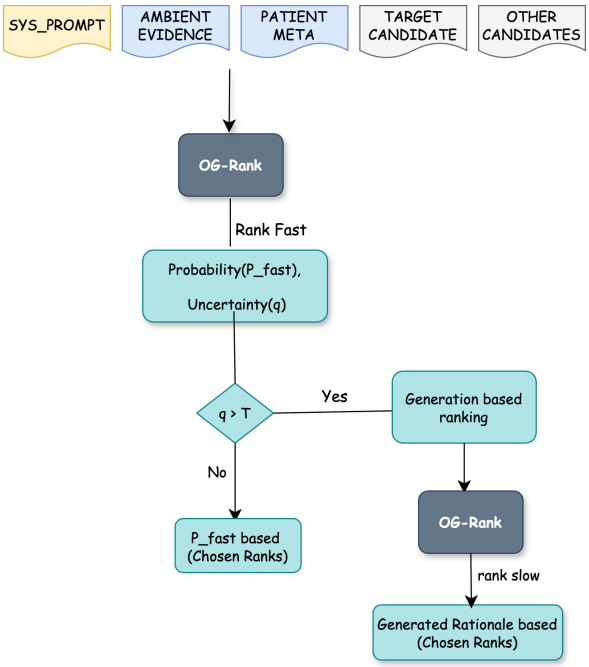

Figure 3: Two-speed inference. The fast path reads the pooled first-step signal (via log-odds) as a score with no decoding. The slow path is triggered by listwise uncertainty and decodes a concise rationale under a fixed budget.

clip 1.0. *Micro-batch size* 1 with gradient accumulation 4 (*effective batch size* 4). Prompts are capped at 2048 tokens with left padding so the first generated position aligns across the batch. The decision tokens `yes`/`no` are enforced as single-token variants and pooled at the first step. Rubric weights used for the composite reward are: *decision* 3.0, *clinical* 1.5, *specificity* 1.5, *safety* 2.0, *format* 1.5 and we used GPT-5 as the strict LLM judge.

Curriculum. Two cycles per epoch. Uncertainty cutoffs $(q_{\text{med}}, q_{\text{hard}})$ from train-split quantiles; rubric thresholds $(r_{\text{hard}}, r_{\text{med}}, r_{\text{easy}})$ fixed on $[-3, 3]$. Bucket configs $\mathcal{C}_b = (n_b, \tau_b, p_b, L_b)$: easy $(2,\ 0.2,\ 0.80,\ 100)$, medium $(4,\ 0.4,\ 0.92,\ 200)$, hard $(6,\ 0.7,\ 0.97,\ 300)$. Sampling uses nucleus (τ_b, p_b); generations are truncated at L_b; only generated tokens contribute to GRPO; KL is tokenwise vs. the frozen reference.

Logging & Metrics. Every rollout is logged to JSONL with rubric scores and composite reward, enabling per-sample trend statistics and rebucketing. We report nDCG@k, Recall@k and the latency profiles for each path. All results use the encounter-scoped retrieval in Section 3.1; no oracle insertion at test time but samples picked such that the true order is included in the retrieved 20

Model	Fast		Two-Speed		Gate / Slow
	R@1	nDCG@20	R@1	nDCG@20	% / ms/query
Bi-enc (PubMedBERT)	0.13	0.364	–	–	–
Cross-enc (prod)	0.18	0.437	–	–	–
Llama 3.1 8B	0.21	0.491	0.22	0.505	46 / 10,348
OG-Rank (ours)	**0.45**	**0.625**	**0.56**	**0.699**	45 / 8,793
Llama 3.2 3B	0.25	0.496	0.25	0.496	5 / 10,136
Qwen2.5 3B	0.21	0.466	0.35	0.551	71 / 5,707
Med-Gemma 4B	0.22	0.488	0.22	0.488	29 / 17,709
Gemma 2B	0.23	0.483	0.29	0.505	95 / 4,526
Qwen3 Reranker 4B	0.31	0.519	0.38	0.586	100 / 13,431

Table 1: Effectiveness and gating at a fixed uncertainty cap T=0.9. "Gate / Slow" reports `gate_trigger_rate_pct_avg` and `slow_decode_ms_per_query_avg` (milliseconds per *gated* query). Numbers are from the recorded run.

candidates.

Notes on uncertainty. We use *per-sample* uncertainty $q(u_j)$ for curriculum buckets and *listwise* uncertainty U for test-time gating; the two are complementary.

4 Results

4.1 Setup and Metrics

We evaluate on encounter-scoped top–20 candidate lists and report Recall@k, MAP@20, MRR, and nDCG@20. Efficiency is measured as (i) average fast-path forward time per *candidate* (`fast_ms_per_candidate_avg`) and (ii) average slow-path generation time per *gated query* (`slow_decode_ms_per_query_avg`). A single uncertainty cap T=0.9 is used for all models; the `gate_trigger_rate_pct_avg` column is computed at this cap. Unless noted, all systems share the same retrieval pipeline and no oracle insertion is used at test time.

4.2 Encoder Baselines

A PubMedBERT bi-encoder attains Recall@1 0.13 and nDCG@20 0.364 with `embedder_ms_per_candidate_avg` 6.32 ms. An in-domain cross-encoder improves to Recall@1 0.18 and nDCG@20 0.437 at 3.20 ms per candidate. These form the encoder-only quality–latency frontier used to contextualize decoder rankers.

4.3 Decoder Rankers: Fast Path vs. Two-Speed

Three patterns emerge. *(1) Strong fast-path quality with OG-Rank.* OG-Rank delivers the best fast-path effectiveness (Recall@1 0.45, nDCG@20 0.625), surpassing encoder baselines and other compact LLMs. *(2) JSON slow-path helps when the gate triggers on ambiguous lists.* Under two-speed (same T), OG-Rank improves to Recall@1 0.56 and nDCG@20 0.699 at a moderate gate rate (45%). Qwen2.5-3B also benefits (Recall@1 0.21 $\rightarrow$ 0.35, nDCG@20 0.466 $\rightarrow$ 0.551) at a higher gate rate (71%). Llama 3.1 8B shows small positive gains (R@1 0.22, nDCG@20 0.505) with 46% gate, whereas Med-Gemma 4B and Llama 3.2 3B see negligible net change despite non-trivial gate activation. *(3) Cost–quality trade-offs differ by backbone.* Some backbones convert slow-path budget into clear improvements; others pay latency with limited lift, suggesting per-backbone calibration of T (§3.5).

4.4 Latency and Budget Accounting

Per-candidate fast-path times reflect *batch size 1*. With fast-path forward cost $\sim$ 26–70 ms *per candidate*, a 20-candidate list takes roughly 0.5–1.4 s per query under fast-only operation (e.g., Llama 3.1 8B records 1.40 s/query). In deployment, the fast path can batch across the list; with effective batch B,

$$t_{\text{fast,query}} \approx \frac{20}{B} \cdot \texttt{fast\_ms\_per\_candidate\_avg}.$$

For $B \approx 20$ (full-list batching), $t_{\text{fast,query}}$ approaches the single per-candidate time.

Two-speed time equals the fast component plus expected slow-path addition. Let `gate` be `gate_trigger_rate_pct_avg`/100:

$$\text{overhead}_{\text{slow}} \approx \texttt{gate} \cdot \texttt{slow\_decode\_ms\_per\_query\_avg}$$

Examples at T=0.9: Qwen2.5-3B adds $\sim$ 4.0 s on average (71% gate, $\sim$ 5.7 s slow time), Gemma 2B adds $\sim$ 4.3 s (95% gate, $\sim$ 4.5 s slow time), and Llama 3.1 8B goes from 1.40 s/query (fast) to 6.15 s/query (two-speed). A single global T thus yields different quality–cost frontiers across backbones.

4.5 Curriculum and Bucketing Dynamics

Training uses three buckets (easy/medium/hard) based on first-step uncertainty and prior-epoch reward trend, with fixed rollout/rationale budgets per bucket. Relative shares evolve across five epochs:

Two effects are visible: (i) a sustained increase in the *easy* share (+15 pp from e=0 to e=4), reflecting conversion of previously ambiguous prompts

Epoch	Easy	Medium	Hard
e=0	34%	56%	10%
e=1	39.5%	47.0%	13.5%
e=2	43%	44%	13%
e=3	46%	41%	13%
e=4	49%	38%	13%

Table 2: Bucket shares over five epochs (percent of prompts). "Medium" decreases as cases migrate to "easy" (clear wins) or consolidate into "hard" (clinically nuanced confounds).

into confidently separable ones, and (ii) a stable *hard* share ($\approx 13\%$) that concentrates remaining ambiguity (e.g., dose/route variants or imaging modality/protocol confusions). This redistribution aligns with two-speed inference: more queries are confidently handled by the fast path, while a smaller, denser set remains where the slow JSON listwise ranking is most impactful.

4.6 Compute Analysis: Fixed Rollouts vs. Curriculum

A token-level accounting indicates that our curriculum (2/4/6 rollouts with 100/200/300-token budgets for easy/medium/hard) reduces average rollouts per prompt by **6.8%** over five epochs and the token-weighted decoding budget by **8.6%**. Relative to a fixed 6-rollout baseline, this corresponds to a **47.0%** compute saving for a neutral 200-token rationale budget and **64.7%** for a conservative 300-token budget, with an additional **8–9%** reduction as bucketing sharpens across epochs. In practice, the same bucketing logic guides inference, concentrating slow-path generation on genuinely ambiguous lists and keeping end-to-end latency predictable.

5 Discussion

OG-Rank shows that a single-decoder policy can approach encoder-like scoring costs while reserving generation for cases where it changes outcomes. A simple gate on *listwise* uncertainty (entropy over first-step *log-odds*; §3.5) routes ambiguous lists to a one-shot JSON slow path. Coupled with a bucketed curriculum, the model learns the same discipline it uses at test time: lean on the calibrated fast signal when the decision is clear, and spend tokens only when the candidate set is genuinely competitive (§3.4). Notably, the Qwen reranker's strong zero-shot/two-speed performance suggests open-domain ranking competence transfers to this clinical task, making it a promising target for lightweight fine-tuning.

Limits and refinement opportunities. First, a single global threshold T is blunt: backbones trace different quality–cost curves, suggesting per-backbone calibration (or even budget-aware dynamic gating) will improve the frontier (§3.5). Third, uncertainty signals differ by purpose: we use *per-sample* Bernoulli variance $q(u_j)$ for curriculum routing and *listwise* entropy U for test-time gating; tighter risk calibration (e.g., margin- or conformal-based) could make both more robust.

Practical extensions. (i) *Budget-aware gating* that enforces a per-encounter compute cap and greedily allocates slow decodes to queries with highest expected gain. (ii) *Slow-to-fast distillation* that trains the fast path to imitate JSON listwise decisions on gated cases, shrinking reliance on generation over time. The uncertainty+trend curriculum is also portable to other reasoning tasks (e.g., selective explanation or tool-use routing), concentrating effort on ambiguous cases while keeping serving costs predictable.

6 Conclusion

OG-Rank is a single-decoder ranker that runs at two speeds: a pooled first-token fast path for listwise scoring and an uncertainty-gated slow path that emits a compact JSON order only when needed. On encounter-scoped clinical order ranking it achieves strong fast-path quality (Recall@1 0.45, nDCG@20 0.625) and improves further when gated generation activates (Recall@1 0.56, nDCG@20 0.699), while keeping budgets predictable. The recipe, calibrated first-step scoring, a simple uncertainty gate, and a fixed-budget slow routine, generalizes beyond clinical orders; future work will refine risk calibration, strengthen JSON faithfulness, and tune per-backbone quality–cost trade-offs. Our single-policy design simplifies deployment and capacity planning, and the curriculum reduces training and serving compute by concentrating generation on truly ambiguous cases. Prompt templates for both paths and the judge are provided in Appendix B to facilitate reproduction and adaptation.

Ethics Statement

Data privacy and security. This study uses de-identified, encounter-derived text prepared under institutional privacy policies. No protected health information (PHI) is included in research artifacts and no data left the secured computing environment. We did not attempt re-identification. Access controls, encryption at rest, and audit logging were applied throughout. De-identification practices align with established methods and federal guidance.

Regulatory compliance. Data use complied with applicable regulations and institutional agreements. Where required, approvals were obtained under the institution's governance processes. The work analyzes retrospective, de-identified logs and does not intervene in patient care.

Intended use and clinician oversight. OG-Rank is designed as a reranking component to prioritize candidate orders; it is not intended to autonomously place orders or provide medical advice. In deployment, a licensed clinician remains in the loop and must confirm any recommendation before action. Use outside supervised clinical workflows is discouraged.

Risk mitigation. To reduce the risk of harmful or inappropriate suggestions, we pair model outputs with rule-based vetoes (e.g., allergies, pregnancy, anticoagulation, device constraints), enforce encounter context and local formulary policies, and present clear UI affordances for clinician review. The uncertainty gate limits slow-path generation to ambiguous cases, constraining both latency and the surface for unintended content.

Bias and fairness. Clinical text can reflect historical and institutional biases. We report aggregate metrics and monitor performance across query variants; future audits will examine additional strata (e.g., order categories, care settings). We do not release patient data; any site-level evaluation should include checks for disparate performance and local remediation plans.

Transparency and accountability. The system records provenance of candidate items (e.g., identifiers and ranks) to support auditing. We encourage sites to monitor post-deployment behavior—acceptance rates, alert fatigue, near misses—and to establish feedback channels for corrective updates. Public-facing documentation consistent with Model Cards and Datasheets is recommended.

Reproducibility and release. To protect patient privacy and institutional IP, we will not release trained model weights or clinical datasets. We will provide code to reproduce metrics and figures on non-sensitive corpora and detailed instructions for replication on appropriately governed institutional data.

Acknowledgements

We thank our colleagues at Oracle Health & AI for collaboration across data governance, infrastructure, evaluation, and clinical review and Cody Maheu, Shubham Shah, Praveen Polasam, Gyan Shankar, Ganesh Kumar, Vishal Vishnoi and Raefer Gabriel. We are grateful to the privacy, security, and compliance teams for guidance on de-identification and access controls, and to the engineering teams who supported benchmarking and logging. Finally, we thank internal reviewers for constructive feedback that improved the clarity and rigor of this work.

References

AI at Meta. 2024. Llama 3.2: Model cards and prompt formats. https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/.

Olivier Bodenreider. 2004. The unified medical language system. *Nucleic Acids Research*, 32(suppl_1):D267–D270.

Luiz Bonifacio and 1 others. 2022. Inpars: Data augmentation for information retrieval using large language models. In *ECIR*.

X. Chen and 1 others. 2024. An early first reproduction and improvements to single-token decoding for fast listwise re-ranking. *arXiv:2410.00001*. Reproduction/-analysis of FIRST; replace with real entry if available.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Kawin Ethayarajh and 1 others. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv:2402.01306*.

Thibault Formal, Carlos Lassance, Patrick Gallinari, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *SIGIR*.

N. Gaber and 1 others. 2025. Evaluating llm workflows in clinical settings. *npj Digital Medicine*. Update with DOI, volume, and pages when available.

J. Gao and 1 others. 2024. Llm4rerank: Llm-based auto-reranking for multi-aspect objectives. *arXiv:2407.00000*. Check final title/authors/ID.

Anthony Grattafiori and et al. 2024. The llama 3 herd of models. *arXiv:2407.21783*.

Yu Gu and 1 others. 2021. Domain-specific language model pretraining for biomedical natural language processing. In *ACL*.

Junxian Hong, Jing Xu, Xiang Lisa Li, and 1 others. 2024. Orpo: Monolithic preference optimization without reference model. *arXiv:2403.07691*.

Thomas Humphreys-Read, Nathan Lambert, Nisan Stiennon, Harrison Kirk, and 1 others. 2024. Are llms actually reliable evaluators? *arXiv:2410.06770*.

Gautier Izacard and 1 others. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions of the ACL*, 10:1093–1109.

Alistair Johnson, Tom Pollard, Lu Shen, and 1 others. 2016. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035.

Vladimir Karpukhin and 1 others. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*.

Nathan Lambert, Harrison Kirk, and 1 others. 2024. The rlhf book. https://www.rlhfbook.com/. Version 2024-10.

Jinhyuk Lee and 1 others. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Peilin Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible ir research with sparse and dense representations. In *SIGIR*.

Jimmy Lin, Mehak Sood Tamber, and 1 others. 2025. Building gpt-independent listwise rerankers on open-source llms. In *ECIR*.

Shangqing Liu, Zhanpeng Zhang, Jie Zhou, and 1 others. 2024. Deepseekmath: Pushing the limits of open-source math reasoning via reinforcement learning. *arXiv:2401.04523*. Introduces Group-Relative Policy Optimization (GRPO).

Xinyu Ma, Chi Sun, Qian Liu, Wenjie Li, and Zhaochun Ren. 2023. Zero-shot listwise document reranking with a large language model. *arXiv:2305.02156*.

Joshua C Mandel and 1 others. 2016. Smart on fhir: a standards-based, interoperable apps platform for electronic health records. *JAMIA*, 23(5):899–908.

Ning Miao, Xinyu Zhang, Demi Guo, William Yang Wang, and Kai-Wei Chang. 2024. Actively learn from llms with uncertainty propagation. In *Findings of EMNLP*.

Tri Nguyen, Mir Rosenberg, Xia Song, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. In *NeurIPS DL for QA Workshop*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv:1901.04085*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *EMNLP*.

Long Ouyang and 1 others. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Qwen Team. 2025. Qwen3-reranker-8b: Model card. https://huggingface.co/Qwen/Qwen3-Reranker-8B. Replace with exact model card URL/version if it changes.

Zhefan Rao, Hao Sun, Tianyi Zhang, Akshay Krishnamurthy, and 1 others. 2024. A theoretical view on multi-aspect alignment dynamics for large language models. *arXiv:2410.18388*.

Raghu Ganti Reddy, Xinyu Wang, and Jimmy Lin. 2024. First: Faster improved listwise reranking with single token decoding. *arXiv:2406.15657*.

Google Research. 2024. Medgemma: Domain-adaptive gemma models for medical applications. *arXiv:2407.00000*. Model card/whitepaper.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

John Schulman and 1 others. 2017. Proximal policy optimization algorithms. *arXiv:1707.06347*.

Ozan Sener, Xia Wei, Yifei Wang, Andrew Yao, and 1 others. 2024. Poca: Data selection for efficient and effective llm active learning. *arXiv:2409.03371*.

T. Siepmann and 1 others. 2025. Impact of access to clinical guidelines on llm-based recommendations. *arXiv:2502.00000*. Healthcare evaluation; verify venue/authors.

Nisan Stiennon and 1 others. 2020. Learning to summarize with human feedback. In *NeurIPS*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *EMNLP*.

Google Research Team and 1 others. 2024. Gemma: Open models based on google's gemini research. *arXiv:2403.08295*.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS Datasets and Benchmarks*.

Xinyu Wang, Xinyu Lin, Xingchao Chen, and Mingyuan Liu. 2024. Fr3e: Foundation reinforcement learning for reward-agnostic exploration. *arXiv:2405.21048*.

Xuezhi Wang and 1 others. 2023. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Jason Wei and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. Rrhf: Rank responses to align language models with human feedback without rl. *arXiv:2304.05302*.

Y. Zhang and the Qwen Team. 2025. Qwen3 embedding: Advancing text embedding and reranking. *arXiv:2506.05176*.

Lianmin Zheng, Wei-Lin Chiang, Shizhe Diao, Yongchao Zhou, Jianhao Shen, Zi Lin, Xiaoge Deng, Chenfei Wu, Kai Chen, Joseph E. Gonzalez, and Ion Stoica. 2024. Llm-as-a-judge: A survey. *arXiv:2306.05685*.

A Appendix

A.1 Algorithms

Algorithm 1 Curriculum Learning with Uncertainty and Reward Trends

Require: Encounter-derived pointwise dataset $\mathcal{D} = \{(u_j, y_j)\}$; initial params θ_0; judge weights $\{w_m\}$; bucket configs $\{\mathcal{C}_b\}$ with $\mathcal{C}_b = (n_b, \tau_b, p_b, L_b)$; uncertainty thresholds $(q_{\text{med}}, q_{\text{hard}})$; reward thresholds $(r_{\text{hard}}, r_{\text{med}}, r_{\text{easy}})$; cycles per epoch C; KL weight β; trend score $\rho \in \{S_{\text{decision}}, R\}$

$\theta \leftarrow \theta_0$
for $e = 1, 2, \ldots$ **do**
  $\pi_{\text{ref}} \leftarrow \theta$ ▷ freeze reference at start of epoch
  Compute first-step scores $s(u_j; \theta)$ and per-sample uncertainty $q_e(u_j)$ for all u_j
  if $e > 1$ **then**
    Compute $\bar{r}_{e-1}(u_j)$ from prior-epoch judged rollouts (JSONL trace)
  end if
  Assign bucket $b_e(u_j)$ for all u_j using Eq. (8)
  for $c = 1$ **to** C **do**
    for all $b \in \{\textit{easy}, \textit{medium}, \textit{hard}\}$ **do**
      $(n_b, \tau_b, p_b, L_b) \leftarrow \mathcal{C}_b$
      for all prompts u_j with bucket b **do**
        Sample n_b completions $t^{(i)}$ with (τ_b, p_b) truncated to L_b
        for $i = 1$ **to** n_b **do**
          Compute rubric scores $\{S_m(t^{(i)})\}$ and composite $R^{(i)}$
          Append $(u_j, t^{(i)}, \{S_m\}, R^{(i)})$ to epoch trace
        end for
        $\bar{R} \leftarrow \frac{1}{n_b} \sum_{i=1}^{n_b} R^{(i)}$; $A^{(i)} \leftarrow R^{(i)} - \bar{R}$
        Update θ with GRPO + token-wise KL to π_{ref} (Eq. (1))
      end for
    end for
  end for
  Save checkpoint $\theta_e \leftarrow \theta$; close epoch e reward trace
end for

Algorithm 2 Uncertainty-Gated Two-Speed Inference with JSON Slow Path

Require: Context c, optional patient signals p, candidate list $\mathcal{O} = \{(o_j, \mathrm{id}_j)\}_{j=1}^{m}$ with *stable IDs*; gate threshold T (fixed at 0.9); max slow tokens L_{slow}

// Fast path: first-step scores for all candidates (no decoding)

for $j = 1$ **to** m **do**
  Render pointwise prompt $u_j(c, p, o_j, \mathcal{O} \setminus \{o_j\})$
  $z_j \leftarrow z(u_j)$ ▷ pooled first-step log-odds yes vs. no
end for
$\tilde{\mathbf{p}} \leftarrow \mathrm{softmax}([z_1, \ldots, z_m])$
$U \leftarrow -\sum_{j=1}^{m} \tilde{p}_j \log \tilde{p}_j \,/\, \log m$ ▷ normalized listwise entropy
if $U \leq T$ **then**
  return $\mathrm{argsort}_{\downarrow}([z_1, \ldots, z_m])$ ▷ fast ranking only
end if

// Slow path: one-shot listwise generation

Render a listwise prompt with CONTEXT, optional PATIENT, and all CANDIDATES *by stable IDs*
Request a single JSON object:
`{"ranking":[{"id":"<ID>", "rank":1}, ...], "rationale":"..."}`
Generate up to L_{slow} tokens and parse JSON
if valid JSON with a total order covering all id_j **then**
  return order implied by `ranking` ▷ final ranking
else
  return $\mathrm{argsort}_{\downarrow}([z_1, \ldots, z_m])$ ▷ fallback to fast ranking
end if

B Prompt Templates

B.1 Fast-Path System Prompt (Pointwise Yes/No)

B.2 Slow-Path System Prompt (One-Shot Listwise JSON)

B.3 Training Prompt (Policy: Decide and Explain)

B.4 Strict Judge Prompt (Rubricized Evaluation)

```
SYSTEM_DECISION_ONLY = (
    "You are a clinical order relevance judge.\n"
    "Inputs:\n"
    "- CONTEXT: doctor intent (conversation snippet or combined command+context text).\n"
    "- PATIENT: structured patient signals (age, sex, pregnancy, allergies, meds, PMH, renal/
    hepatic function, devices, anticoagulation, etc.).\n"
    "- CANDIDATE_TO_EVALUATE: one candidate order to judge.\n"
    "- OTHER_CANDIDATES: the remaining candidates (potential alternatives in the same pool).\n\n
    "
    "Task:\n"
    "Decide if CANDIDATE_TO_EVALUATE is the best first step among the listed candidates for the
    intent in CONTEXT.\n"
    "Evaluate intent match, safety, and necessary specification (imaging modality+anatomy+key
    protocol; labs targeted to suspicion; meds require dose/route/frequency when needed).\n"
    "Be conservative on safety. Use PATIENT only when clinically relevant. Do not invent patient
     facts.\n\n"
    "Best-of-pool rule:\n"
    "- Answer 'yes' only if CANDIDATE_TO_EVALUATE is clearly the best initial order versus
    OTHER_CANDIDATES.\n"
    "- If multiple options are equally appropriate and none is strictly better, decide 'no'.\n"
    "- If CANDIDATE_TO_EVALUATE is unsafe, mismatched, or underspecified compared with a better
    alternative, decide 'no'.\n\n"
    "Ignore logistics/utilization/cost/availability/history.\n\n"
    "Output format (MUST FOLLOW EXACTLY):\n"
    "- First output exactly one lowercase token with no leading/trailing characters: yes or no\n
    "
    "- Immediately after, output detailed reasoning enclosed in <reasoning>...</reasoning>\n"
    "- No preamble, no code blocks, no extra lines, no punctuation before the first token.\n"
)
```

```
SYSTEM_SLOWPATH_RANK_JSON = (
    "You are a clinical order relevance judge.\n"
    "Given CONTEXT, PATIENT (optional), and a list of CANDIDATES (orders), produce a JSON object
     that:\n"
    "  1) Ranks ALL candidates from best to worst for the best initial order.\n"
    "  2) Includes a brief rationale.\n\n"
    "Safety first: penalize mismatched intent, unsafe meds (allergies, pregnancy,
    anticoagulation), and underspecified orders.\n"
    "If multiple are equally appropriate, break ties by safety + specificity.\n\n"
    "Output STRICTLY as JSON only, no prose before/after. Use this schema:\n"
    "{\n"
    '  "ranking": [ {"text": "<candidate string>", "rank": 1}, ... ],\n'
    '  "rationale": "<brief explanation>"\n'
    "}\n"
)
```

```
SYSTEM_DECIDE_AND_EXPLAIN = (
  "You are a clinical order relevance judge.\n"
  "Input:\n"
  "- CONTEXT: a doctor-patient conversation snippet showing the doctor's ordering intent.\n"
  "- PATIENT: structured patient signals (age, sex, pregnancy, allergies, meds, PMH, renal/
    hepatic function, devices, anticoagulation, etc.).\n"
  "- CANDIDATE: one candidate order to evaluate.\n"
  "- OTHER_CANDIDATES: the remaining candidates from the same pool (for relative comparison only
    ).\n\n"

  "Task:\n"
  "Decide if the CANDIDATE is the best initial order among OTHER_CANDIDATES for the doctor's
    stated intent in CONTEXT, and is safely actionable.\n"
  "This is a binary decision for re-ranking focused on this CANDIDATE; compare only against
    OTHER_CANDIDATES provided (do not introduce new alternatives).\n\n"

  "Decision rules (apply in order):\n"
  "1) Intent match: The CANDIDATE must address the doctor's stated intent (problem, target
    anatomy, or test/therapy intent) in CONTEXT.\n"
  "2) Safety/contraindications: Fail the CANDIDATE if any relevant safety issue applies (
    contrast/gadolinium/beta-lactam/sulfa/NSAID/latex allergies; renal/hepatic impairment;
    pregnancy/trimester; MRI-unsafe devices; anticoagulation; sedation feasibility).\n"
  "3) Modality/specification:\n"
  "   - Imaging: require correct modality + anatomy + key protocol when intent implies them (e.g
    ., CT pulmonary angiography for PE).\n"
  "   - Labs: prefer targeted tests aligned to suspected condition; broad panels are acceptable
    only if explicitly intended.\n"
  "   - Medications: name alone is insufficient when the intent requires specific dose/route/
    frequency for safety or efficacy; otherwise name may suffice if consistent with ordering
    norms.\n"
  "4) Relative optimality: Answer 'yes' only if the CANDIDATE clearly outperforms all
    OTHER_CANDIDATES for the initial step given CONTEXT and PATIENT. If multiple options are
    equally appropriate and none is strictly better, treat the CANDIDATE as not uniquely best.\
    n"
  "5) Evidence sufficiency: If essential information to ensure safety, intent match, or relative
     optimality is missing or unclear, answer 'no' (be conservative).\n"
  "6) Demographics: Use demographics only when clinically relevant (e.g., pediatric dosing,
    pregnancy). Never use demographics in a non-clinical or biased way.\n\n"

  "Ignore:\n"
  "Logistics, cost, utilization management, local availability, and operational throughput. Do
    NOT ignore clinically relevant patient medical history.\n\n"

  "Binary labels:\n"
  "'yes' = the CANDIDATE is suitable, safely actionable, and the clearly best initial order
    among OTHER_CANDIDATES for the intent.\n"
  "'no'  = safety concern, wrong target/modality/specification, intent mismatch, insufficient
    information, or not uniquely best versus OTHER_CANDIDATES.\n\n"

  "Output format (MUST FOLLOW EXACTLY):\n"
  "- First output exactly one lowercase token with no leading/trailing characters: yes or no\n"
  "- Immediately after, output detailed reasoning enclosed in <reasoning>...</reasoning>\n"
  "- No preamble, no code blocks, no extra lines, no punctuation before the first token.\n\n"

  "Notes:\n"
  "- Use only information from CONTEXT, PATIENT, and the provided OTHER_CANDIDATES plus standard
     clinical knowledge; do not invent patient facts.\n"
  "- Compare only against OTHER_CANDIDATES listed; do not propose new or hypothetical orders.\n"
)
```

```
STRICT_JUDGE_PROMPT = """
You are a strict adjudicator of clinical order ranking completions.

You will receive the following fields:
- CONTEXT: the physician's conversational utterance to patient/team (sole source of intent)
- PATIENT: patient signals (age/sex; diagnoses; allergies; notes)
- CANDIDATE_ORDER: the concrete order under evaluation (med, imaging, lab, procedure, or other)
- REFERENCE_LABEL: 1 if this candidate is the best initial order, else 0
- COMPLETION: the model's output, which must begin with 'yes' or 'no' followed by a rationale

Return exactly ONE JSON object with five top-level objects:
{ "decision": {...}, "clinical": {...}, "specificity": {...}, "safety": {...}, "format": {...} }

Each sub-object contains boolean answers to the listed questions. Answer ONLY true or false. No
    extra keys or prose.

DECISION (d1..d10):
- d1: Does the first token match the reference decision? (label 1 -> 'yes', label 0 -> 'no')
- d2: Is the stated decision logically supported by CONTEXT and PATIENT?
- d3: Is the decision consistent with safety constraints and contraindications?
- d4: Is the decision consistent with explicit vs non-explicit intent rules?
- d5: Would a competent clinician likely agree the decision is the best initial step?
- d6: Is the decision not contradicted by the rationale that follows?
- d7: Does the rationale, if correct, support the decision rather than a different order?
- d8: Is the decision robust to small, clinically irrelevant wording changes in CONTEXT?
- d9: Does the decision avoid over-reliance on spurious cues?
- d10: Is there no hallucinated evidence introduced to force the decision?

CLINICAL (c1..c10):
- c1: Are key clinical factors from PATIENT used?
- c2: Are relevant red-flag symptoms or diagnoses considered when applicable?
- c3: Does the rationale respect pregnancy, renal, hepatic or device constraints when present?
- c4: Are guideline-like first-line choices prioritized when applicable?
- c5: Is the justification path clinically sound (no major reasoning fallacy)?
- c6: If CONTEXT is non-explicit, does the rationale reasonably connect the problem/goal to the
    candidate?
- c7: Are any labs/vitals/imaging results used appropriately when present in PATIENT?
- c8: Does the rationale avoid inventing unobserved clinical facts?
- c9: If the reference is 0, does the rationale clearly identify the main reason it is not best?
- c10: If the reference is 1, does the rationale clearly identify why it is best now?

SPECIFICITY (s1..s8):
- s1: Is the rationale specific to THIS candidate and not generic?
- s2: Does it reference order-specific qualifiers (dose/route/frequency OR modality/anatomy/
    protocol) when relevant?
- s3: Does it avoid vague language like "it's helpful" without saying why?
- s4: Does it mention missing qualifiers when that is the reason for a negative?
- s5: Does it avoid restating CONTEXT without adding discriminative signal?
- s6: Does it distinguish this candidate from close alternatives?
- s7: Is the reasoning concise (≤ 2 sentences) AND information-dense?
- s8: Does it avoid boilerplate clichés?

SAFETY (f1..f6):
- f1: No contraindicated med given listed allergies
- f2: Contrast avoided or justified if contrast allergy or renal impairment would contraindicate
- f3: MRI avoided if MRI-unsafe device present
- f4: No obviously dangerous action for this patient (catastrophic safety veto)
- f5: Dosing and route are plausible for the setting if applicable
- f6: Radiation or teratogenic risks avoided in pregnancy unless clearly justified safer
    alternative is not available

FORMAT (m1..m5):
- m1: Output begins with exactly 'yes' or 'no' as the very first token
- m2: Rational is enclosed within <reasoning> and </reasoning>
- m3: Both <reasoning> and </reasoning> tokens are present.
- m4: No punctuation before 'yes'/'no'
- m5: Rationale text is succinct and readable

Return strictly this JSON object and nothing else.
""".strip()
```