CORE: DISCOVERING INTRINSIC RANKING PREFERENCES IN LLMS VIA CONSISTENT EGO-CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are powerful listwise rerankers, but their performance is notoriously sensitive to prompt variations, undermining their reliability for real-world applications. To address this, we proposes CORE, a new fine-tuning framework that mitigates this instability by learning a model's intrinsic, prompt-invariant ranking preferences. CORE integrates two complementary mechanisms: a guidance strategy adapted from Classifier-Free Guidance to calibrate the generative process against stylistic variations, and a consistency loss based on differentiable Kendall's Tau to regularize the model's internal ordinal judgments. On standard TREC Deep Learning and BEIR benchmarks, CORE establishes new state-of-the-art ranking performance. Crucially, it demonstrates superior robustness, reducing performance variance across diverse prompts by over 80% compared to standard fine-tuning. Our work presents a principled and effective method for building powerful and trustworthy LLM-based reranking systems.

1 INTRODUCTION

Large Language Models (LLMs) have recently emerged as powerful components in information retrieval (IR) and document ranking systems (Sun et al., 2023b; Long et al., 2025; Gao et al., 2025; Zhuang et al., 2024b). Due to their strong semantic understanding, reasoning, and generation capabilities, LLMs can be adapted to ranking tasks via flexible prompting paradigms (zero-shot, few-shot) or fine-tuning (supervised tuning, direct preference optimization) (Sun et al., 2025), often showing potential beyond traditional neural rankers. LLMs can assess relevance with a nuance that traditional, sparse-vector or dense-vector models often miss (Ma et al., 2023; Sun et al., 2023a). This allows them to capture subtle semantic relationships between a query and a document, making them an invaluable final-stage component in modern search pipelines.

LLM ranking approaches can be broadly categorized into pointwise, pairwise, listwise, and setwise paradigms (Sun et al., 2025; Long et al., 2025; Zhuang et al., 2024b). These differ in how the LLM processes relevance signals. In a pointwise approach (Sachan et al., 2022; Zhuang et al., 2024a; Fan et al., 2025), the LLM evaluates each document's relevance to the query independently. For instance, an LLM may be prompted to output a binary "Yes/No" or a score indicating whether a given document is relevant to the query, and each document is ranked by this score. This paradigm is straightforward and allows parallel scoring of documents, but it ignores inter-document comparisons. In a pairwise approach (Qin et al., 2024; Chen et al., 2025), the LLM compares two documents at a time to decide which is more relevant (e.g. prompting "Which document, A or B, is more relevant to the query?"). Repeating such comparisons across document pairs can yield a preference-based ranking. Pairwise LLM rankers such as PRP (Qin et al., 2024) tend to achieve high accuracy by directly modeling comparative relevance, at the cost of requiring many LLM inference calls (quadratic in the number of documents). In a listwise approach (Ren et al., 2025; Liu et al., 2025), the LLM considers the entire set of candidate documents and produces a sorted list in one go. Listwise prompting can capture complex dependencies between documents (such as diversity or redundancy) and fully leverage the LLM's generative ability to output an ordered list. Early works like RankT5 (Zhuang et al., 2023) showed the feasibility of sequence-to-sequence ranking, and more recent zero-shot systems like RankGPT (Sun et al., 2023b) and RankVicuna (Pradeep et al., 2023b) have demonstrated strong listwise re-ranking performance using GPT-4 and Vicuna-13B respectively.

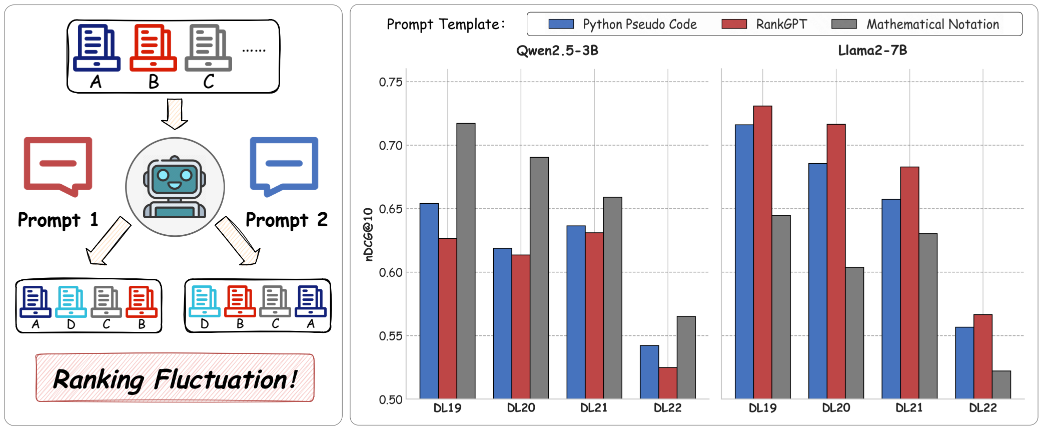

Figure 1: The problem of prompt sensitivity in LLM rerankers. The left panel provides a conceptual illustration: for the same set of documents, two semantically equivalent but stylistically different prompts can lead to disparate ranking outcomes, a phenomenon we term "Ranking Fluctuation". The right panel presents empirical evidence, showing that the nDCG@10 performance for both Qwen2.5-3B and Llama2-7B in the zero-shot setting varies significantly across three different prompt templates on the TREC DL datasets, highlighting the severity of this issue.

Despite the superior performance of LLM-based rankers, the reliability of LLM-based rerankers is severely undermined by a critical weakness: prompt sensitivity. LLMs are notorious for their susceptibility to prompt wording and format – seemingly minor differences in how the query and instructions are phrased can lead to significant changes in the output ranking (Chatterjee et al., 2024; Sclar et al., 2024a; Arabzadeh & Clarke, 2025). For an identical query and document set, subtle, semantically irrelevant variations to the prompt—such as changes in wording, output format, or even the initial order of documents—can lead to dramatically different ranked lists. This instability manifests as significant positional biases and a stark lack of invariance to input permutations (Tang et al., 2024; Sun et al., 2025). We visually demonstrate this problem in Figure 1.

Besides, when controlling for the same LLM, the effectiveness gaps between pointwise, pairwise, listwise, and setwise methods become much smaller once prompt variations are taken into account (Sun et al., 2025). It suggests that some methods appeared better only because they used better prompts. This prompt sensitivity undermines the reliability of LLM rankers: it becomes unclear whether a performance gain is due to a truly better ranking technique or just a better prompt.

To move beyond merely observing this instability and toward a principled solution, we argue that the goal is to uncover the model's *intrinsic preference*—a stable, core ranking capability shielded from superficial prompt variations. Just as a master chef aims to replicate a signature dish's taste consistently regardless of the kitchen's conditions, we seek to distill the LLM's core ranking "palate". We find a powerful analogy for this challenge in Sigmund Freud's structural model of the psyche (Freud, 1923). We metaphorically define the model's latent, intrinsic ranking capability as its **"Id"** (R^*)—its ideal, stable state. In contrast, the actual, observable ranked list produced under a specific prompt p is its **"Ego"** (R_p)—the "Id's" expression under external influence. From this perspective, the observed *ranking fluctuation* is not just a surface-level inconsistency; it reflects a deeper "cognitive gap" between the model's singular "Id" and its multiple, prompt-dependent "Egos". Therefore, the central goal of this work is to resolve this cognitive gap: we aim to learn the robust, prompt-invariant "Id" by treating the inconsistent "Egos" as signals for regularization. We term this process "Consistent Ego-Correction".

To resolve this gap, we propose a novel fine-tuning framework, CORE (COnsistent Reranking via Ego-correction), designed to discover and stabilize the intrinsic preferences of LLM rankers. The framework integrates two key components: **1) External Behavior Calibration via Inverse CFG:** Adapting Classifier-Free Guidance (CFG) (Ho & Salimans, 2022), this mechanism uses a generic, task-defining prompt to anchor the model's response, mitigating biases from specific, stylized user prompts. **2) Internal Judgment Consistency via Differentiable Kendall's Tau:** A novel loss function based on a differentiable Kendall's Tau (Kendall, 1938; Guan et al., 2024) is used to enforce consistent relative rankings of documents across multiple prompt variations. These components compel the model to develop a robust ranking function that is insensitive to superficial prompt phrasing.

The main contributions of this work are summarized as follows:

- We propose a novel cognitive framework (Id-Ego) and provide a formal technical interpretation for it, offering a new perspective for understanding and addressing the prompt sensitivity problem in LLM rankers.
- We design the CORE methodology, which uniquely combines a CFG-based guidance mechanism with an ordinal consistency regularizer, specifically for enhancing the robustness of listwise reranking.
- We conduct extensive empirical evaluations on multiple standard IR benchmarks, showing that CORE not only achieves state-of-the-art ranking effectiveness but, more importantly, exhibits significantly enhanced robustness against prompt variations.

2 Problem Formulation

In this section, we formally define the task of listwise generative reranking, introduce the challenge of ranking fluctuation and present our Id-Ego framework to conceptualize this problem.

2.1 Task Definition: Listwise Generative Reranking

Listwise Document Reranking The task of document reranking is a crucial second stage in modern search systems. Given a user query q and an initial set of N candidate documents $D = \{d_1, d_2, \ldots, d_N\}$ retrieved by a first-stage ranker (e.g., BM25 or a dense retriever), the goal of a reranker f is to produce a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$ of the document indices $\{1, 2, \ldots, N\}$. This permutation should order the documents in descending order of their relevance to the query q. The quality of the final ranked list is typically evaluated using metrics such as Normalized Discounted Cumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002).

LLM-based Generative Reranking Large Language Models are increasingly being employed as powerful listwise rerankers. In this paradigm, the reranking function f is realized by the LLM itself. The query q, the set of candidate documents D, and a task instruction I are formatted into a single textual prompt P using a template function f_{prompt}:

$$P = f_{\text{prompt}}(I, q, D) \tag{1}$$

The LLM, parameterized by θ, then processes this prompt autoregressively to generate the final ranked list π_P. The output is typically a textual sequence of document identifiers, such as "[3] > [1] > [2]", from which the permutation π can be parsed.

$$\pi_P = \text{LLM}_\theta(P) \tag{2}$$

2.2 The Id-Ego Framework for Ranking Fluctuation

A key challenge that undermines the reliability of this paradigm is prompt sensitivity. For any given reranking task, one can construct a set of textually different but semantically equivalent prompt templates $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$. An ideal, robust ranker should exhibit invariance to such superficial changes, producing a consistent output regardless of the specific prompt used. However, LLMs often fail to meet this requirement.

Formally, for two different prompts $P_i, P_j \in \mathcal{P}$, it is frequently observed that:

$$\pi_{P_i} \neq \pi_{P_j} \tag{3}$$

We term this variance in rankings under different but synonymous prompts **ranking fluctuation**. This phenomenon reveals a fundamental lack of robustness and poses a significant barrier to the trustworthy deployment of LLMs in critical ranking scenarios.

To better analyze ranking fluctuation, we distinguish between a model's *intrinsic ranking ability* and its *prompt-conditioned realizations*. For a given query-document set, we posit the existence of an ideal, latent ranking function R^* that is invariant to prompt variations. In practice, however, we only observe a family of outputs $\{R_p\}$, each corresponding to a specific prompt p. These outputs can diverge significantly, highlighting the model's sensitivity to superficial changes in instructions or input order.

For intuition, we use the terms **'Id'** and **'Ego'** as a metaphor: the latent, stable function R^* can be seen as the model's "Id", while each observable R_p is an "Ego', i.e., a surface-level realization influenced by the prompt. Under this view, ranking fluctuation is the manifestation of a **gap** between the hidden, invariant "Id" and the multiple, prompt-dependent "Egos'.

The central challenge is therefore to recover the stable R^* from noisy $\{R_p\}$. Our proposed method, CORE, is designed precisely for this purpose: it aligns the prompt-dependent realizations with the intrinsic ranking function through external calibration and internal consistency objectives.

3 Related Work

LLM Ranking and the Challenge of Prompt Sensitivity. The application of Large Language Models to document ranking has evolved through pointwise, pairwise, and listwise paradigms (Sun et al., 2023b). Pointwise methods assess documents individually, for instance by generating relevance scores or query likelihoods, offering simplicity but ignoring inter-document context (Sachan et al., 2022; Zhuang et al., 2023). Pairwise approaches like PRP improve accuracy by making relative judgments between document pairs, but at a high computational cost (Qin et al., 2024). Listwise methods such as RankT5, RankGPT, and RankVicuna leverage the model's full context window by processing an entire candidate list at once, demonstrating significant performance potential (Sun et al., 2025; 2023b; Pradeep et al., 2023b; Waldo & Boussard, 2024). However, this increased contextual awareness exposes a fundamental vulnerability: extreme sensitivity to the prompt. Minor, semantically irrelevant changes to prompt wording, format, or the initial order of documents can drastically alter ranking outcomes (Chatterjee et al., 2024; Sclar et al., 2024b; Arabzadeh & Clarke, 2025; Hu et al., 2024). This instability is particularly evident as positional bias, where models unfairly favor documents at certain positions, violating the crucial principle of permutation invariance for a true ranker (Tang et al., 2024). This unreliability complicates scientific evaluation, as performance gains may stem from prompt engineering rather than methodological innovation.

Approaches to Mitigating Ranking Instability. Existing research to address this challenge can be divided into two main paradigms. **Inference-time** methods apply post-hoc corrections to the outputs of a fixed model. The most prominent of these is self-consistency, where multiple outputs are generated and aggregated via a voting mechanism (Wang et al., 2023; Zhou et al., 2024). For ranking tasks, this is specifically realized as permutation self-consistency, which aggregates rankings from multiple permuted input lists to neutralize positional bias (Tang et al., 2024). While effective, these methods do not alter the model's intrinsic sensitivity and incur a significant multiplicative increase in inference cost. In contrast, **training-time** enhancements aim to instill robustness directly into the model's parameters. This is a more fundamental approach and includes data-centric strategies, such as augmenting the training data with diverse prompt formats (Ngweta et al., 2025; Wei et al., 2025), and objective-centric strategies, which modify the loss function to explicitly encourage prompt invariance, for instance, through contrastive learning (Qiang et al., 2024).

Positioning CORE: Towards a Robust Prior. Our work, CORE, is a training-time framework that offers a novel and principled approach to instilling intrinsic robustness. Conceptually, our method is grounded in a Bayesian perspective, where we aim to learn a strong **prior** (the model's intrinsic preference, our "Id") that is not easily swayed by the noisy **evidence** of a superficial prompt (the "Ego") (Zhao et al., 2021; Fortuin, 2021; Sam et al., 2024). CORE achieves this through a dual mechanism that directly operationalizes this goal. Its external calibration module is a unique, inverse application of **Classifier-Free Guidance**, a technique from diffusion models (Ho & Salimans, 2022). Its internal consistency module uses a differentiable relaxation of the classic **Kendall's Tau** rank correlation coefficient (Kendall, 1938), which is made feasible by recent advances in differentiable sorting operators (Guan et al., 2024; Zheng et al., 2023). By internalizing robustness during training, CORE produces stable rankings in a single forward pass, providing a more fundamental and efficient solution than post-hoc correction methods.

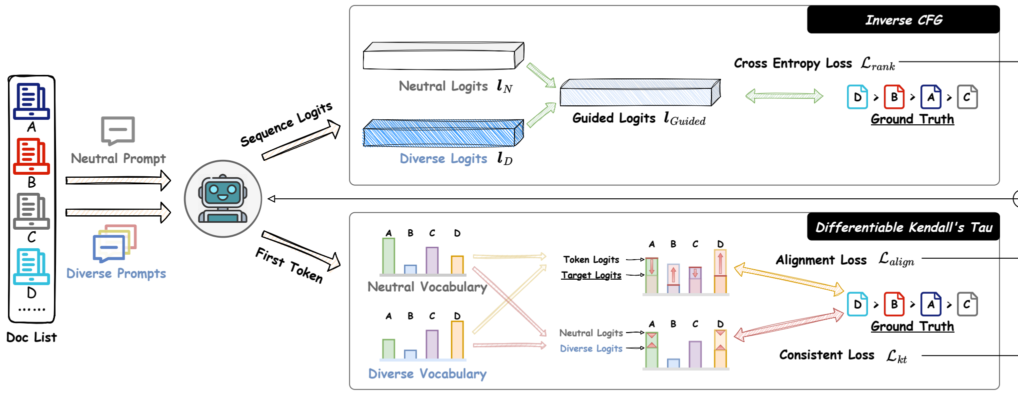

Figure 2: Overview of the proposed CORE framework. It fine-tunes the model through two complementary mechanisms. **Top:** This module calibrates the model's external generative process. It interpolates the logits produced under a diverse prompt with those from a canonical neutral prompt to compute guided logits for the main ranking loss, L_{rank}. **Bottom:** This module regularizes the model's internal ranking judgment. It uses the first-token logits as a differentiable proxy for the overall preference, supervising it for accuracy with an alignment loss, L_{align}, and for stability across prompts with a consistency loss, L_{kt}.

4 METHOD

In this section, we present our methodology, CORE, a fine-tuning framework designed to instill prompt-invariance ability in listwise rerankers. The overall architecture of our framework is illustrated in Figure 2. It resolves the "cognitive gap" between a model's intrinsic ranking capability ("Id") and its prompt-dependent outputs ("Egos") through a dual strategy that we term "external behavior and calibration internal judgment consistency." Externally, we calibrate its step-by-step generative behavior to be less susceptible to stylistic prompt variations. We detail these two complementary mechanisms below. Internally, we enforce consistency on the model's holistic ranking judgment before generation begins.

4.1 EXTERNAL BEHAVIOR CALIBRATION VIA INVERSE CFG

Our first component targets the model's external generative process. The goal is to make the model's final output robust to prompt variations. To achieve this during training, we use two types of prompts: a single, canonical **neutral prompt** ($p_{neutral}$) that represents the pure ranking task, and a set of **diverse prompts** ($\mathcal{P}_{diverse}$) that contain stylistic variations.

This task presents an interesting parallel to the use of Classifier-Free Guidance (CFG) in diffusion models (Ho & Salimans, 2022; Chung et al., 2025). Originally, CFG was designed to *amplify* the influence of a condition to improve stylistic adherence. Its mechanism is formally written as:

$$\hat{\epsilon}_\theta(x_t, c) = \epsilon_\theta(x_t, \emptyset) + w \cdot (\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t, \emptyset)) \tag{4}$$

where $\epsilon_\theta(x_t, c)$ is the model's prediction conditioned on input c, $\epsilon_\theta(x_t, \emptyset)$ is the unconditional prediction, and a guidance scale $w > 1$ steers the final output $\hat{\epsilon}_\theta$ more strongly toward the condition.

Our goal is precisely the opposite: we want to *weaken* the influence of the prompt's stylistic "noise" and steer the model back toward its pure, task-oriented "Id". We thus adapt the CFG principle for this inverse purpose. We treat the output from the noisy $p_{diverse_k}$ as the "conditional output" and the output from $p_{neutral}$ as the "unconditional" baseline.

To formulate this concisely for each generation step j, let $\boldsymbol{l}_{neu}$ and $\boldsymbol{l}_{prm}$ denote the logit vectors produced using the neutral and diverse prompts, respectively. We then compute the calibrated logit vector, $\boldsymbol{l}_{guided}$, by interpolating between them:

$$\boldsymbol{l}_{guided} = (1 - w) \cdot \boldsymbol{l}_{neu} + w \cdot \boldsymbol{l}_{prm} \tag{5}$$

where $w \in [0, 1]$ is the calibration weight. This formulation pulls the potentially biased output from a diverse prompt back toward the stable, neutral baseline. The primary ranking loss, L_{rank}, is a standard listwise cross-entropy loss applied to these guided logits:

$$L_{rank} = -\sum_{j=1}^{N} \log \text{Softmax}(\boldsymbol{l}_{guided,j})[R_{true,j}] \tag{6}$$

This external calibration ensures the model's final output behavior is robustly anchored to the core task, not the prompt's superficial form.

4.2 Internal Judgment Consistency via Differentiable Kendall's Tau

While calibrating the external behavior addresses the final output, we also aim to enforce consistency more directly at the source. Our core motivation is to minimize the ordinal distance between the ranking preferences generated by different prompts. To achieve this, we leverage the Kendall's Tau (τ) correlation coefficient—a natural metric for comparing two ranked lists—as a loss function to regularize the model's *internal ranking judgment*, its holistic, pre-generation assessment of the document list.

Conceptually, Kendall's Tau measures ordinal correlation by comparing the number of concordant pairs (P_c), where items are in the same relative order, against discordant pairs (P_d). The formula for its simplest form, τ_a, is:

$$\tau_a = \frac{P_c - P_d}{\frac{1}{2}N(N-1)} \tag{7}$$

This formula perfectly captures our objective of maximizing ordinal agreement. However, applying it directly to optimize a generative LLM presents two fundamental challenges:

1. **How to extract a differentiable ranking signal?** An LLM produces a ranked list (e.g., "[3] >[1] > [2]') token-by-token. While we can parse this final text to get a discrete ranking to compute τ_a, this process itself is non-differentiable. The path from model weights to a final, sorted textual output involves sequential "argmax" operations, which breaks the gradient flow. This makes it extremely difficult to directly optimize the model based on a loss computed from the final generated order. We need a differentiable proxy for the model's ranking preference.
2. **How to create a differentiable loss function?** The standard Kendall's Tau coefficient, as shown in Equation 7, is inherently non-differentiable due to its reliance on the discrete counting of pairs. This prevents its direct use as a loss function for gradient-based optimization.

Our method systematically addresses these two challenges, as detailed below.

4.2.1 Solution 1: A Differentiable Proxy for Ranking Judgment

To solve the first challenge, we need a differentiable signal that represents the model's ranking judgment. Inspired by the insights from FIRST (Reddy et al., 2024), which showed that the logit distribution of the *first* generated token can serve as a powerful proxy for the model's preference over the entire list, we adopt this technique.

We extract the logits corresponding to each document identifier (e.g., "[doc_1]', "[doc_2]') from this initial token's distribution, forming a vector of preference scores $Z = \{z_1, \ldots, z_N\}$. To ensure these scores are meaningful, we first supervise them to be accurate using a pairwise **Judgment Alignment Loss**, L_{align}:

$$\mathcal{L}_{align} = \sum_{r_i < r_j} \frac{1}{i+j} \log(1 + \exp(z_i - z_j)) \tag{8}$$

where the sum is over all ground-truth pairs where document i is more relevant than j. This loss effectively teaches the model to use the first-token logits to express a correct internal assessment of the document list.

4.2.2 SOLUTION 2: A DIFFERENTIABLE ORDINAL CORRELATION LOSS

Having obtained a differentiable ranking signal Z, we now address the second challenge: the non-differentiability of the Kendall's Tau metric itself. We formulate a differentiable variant, τ_d, that replaces the implicit, non-differentiable sign function used for comparing pairs with the smooth, differentiable hyperbolic tangent ("tanh") function.

This allows us to create a **Judgment Consistency Loss**, $L_{consist}$, that maximizes the correlation between the judgment from a neutral prompt ($Z_{neutral}$) and a diverse prompt (Z_{prompt}):

$$L_{kt} = -\tau_d(Z_{neutral}, Z_{prompt}) \tag{9}$$

where $\tau_d(Z_1, Z_2) = \frac{1}{C_N^2} \sum_{i<j} \tanh\big(k(z_{i,1} - z_{j,1})\big) \cdot \tanh\big(k(z_{i,2} - z_{j,2})\big)$.

4.2.3 COMBINED INTERNAL JUDGMENT LOSS

Finally, we combine the alignment and consistency losses into a single loss term, $L_{consist}$, that holistically supervises the model's internal judgment for both accuracy and consistency:

$$L_{consist} = \alpha L_{align} + \beta L_{kt} \tag{10}$$

where α and β are balancing hyperparameters.

4.3 FINAL TRAINING OBJECTIVE

The complete CORE framework is then trained end-to-end by uniting the external behavior calibration loss (L_{rank}) and the internal judgment consistency loss ($L_{consist}$). The hyperparameters α and β effectively control the balance between all three underlying loss components. The final objective is a straightforward sum:

$$L_{CORE} = L_{rank} + L_{consist} \tag{11}$$

Both mechanisms are active only during training. At inference, the model uses a standard single forward pass, incurring no additional computational cost.

5 EXPERIMENTS

5.1 EXPERIMENT SETUP

Datasets. For fine-tuning, we use ∼40k GPT-4 labeled instances created from 5k queries sampled from MS MARCO (Nguyen et al., 2016), following the setup of (Pradeep et al., 2023a). Each training instance consists of a query and a variable number (≤ 20) of candidate passages that need to be reranked. These automatically labeled pairs serve as supervision to align the model's internal preference signal with ground-truth relevance.

For evaluation, we adopt two categories of benchmarks. First, the TREC Deep Learning tracks (MS MARCO passages), including DL19 (Craswell et al., 2020), DL20 (Craswell et al., 2021) (MS MARCO v1), and DL21 (Craswell et al., 2025a), DL22 (Craswell et al., 2025b) (MS MARCO v2), which are widely used for listwise reranking and allow direct comparison with prior work. Second, we consider a diverse subset of BEIR (Thakur et al., 2021) tasks to assess cross-domain robustness and prompt sensitivity, covering `climate-fever`, `dbpedia-entity`, `fever`, `fiqa`, `hotpotqa`, `nfcorpus`, `scidocs`, `scifact`, and `trec-covid`. Unless otherwise specified, we rerank the top-100 documents retrieved by a first-stage retriever for each query.

Evaluation Metrics. We report **nDCG@10** (Järvelin & Kekäläinen, 2002) as the primary evaluation metric, following common practice in listwise reranking. Since our focus is on the second-stage reranking setting, we always rerank the top-100 documents retrieved by a first-stage retriever and thus do not report retrieval-oriented metrics such as MAP@100. Because large language models exhibit inherent stochasticity and instability, we evaluate each model across multiple runs with temperature fixed at zero, and report the average performance. This procedure ensures fair and stable comparison.

Implementation. We instantiate CORE on a decoder-only LLM and fine-tune it with our CORE method. Training uses mixed precision and gradient accumulation. Our baseline models, denoted as "RankX" (e.g., RankZephyr, RankQwen), are standard supervised fine-tuning (SFT) implementations that follow the methodology of RankZephyr (Pradeep et al., 2023a). Models fine-tuned with our approach are denoted as "CORE_X‘ (e.g., CORE_Qwen). Unless otherwise specified, the base model for both baseline and CORE-finetuned variants is Qwen2.5-3B. For sliding-window listwise decoding, we adopt a window size of 20 and a stride of 10, a setup comparable to prior work (Sun et al., 2023b; Pradeep et al., 2023b;a). At inference time, all models, including CORE, follow the standard **autoregressive decoding process** to ensure a fair comparison. To minimize instability, we unify all evaluations under a single **neutral prompt** ($p_{neutral}$), rather than varying prompt templates. We set the maximum context length to 8192 tokens; when the combined input exceeds this limit, we truncate the input to fit within the window. More specific training details and prompt templates can be seen in the appendix A.1 and appendix A.2.

Baselines. We compare CORE against a comprehensive set of baselines to validate its effectiveness. The comparison includes standard retrievers (**BM25** and **SPLADE++ ED**) to establish a performance floor. Our primary competitors are state-of-the-art open-source listwise rerankers, which represent the direct supervised fine-tuning (SFT) counterparts to our method: **RankVicuna** (Pradeep et al., 2023b), **RankZephyr** (Pradeep et al., 2023a), and **RankQwen**, which we create by applying the RankZephyr methodology to the Qwen base model. To situate our work in the broader landscape, we also include the powerful proprietary model **RankGPT**$_4$ (Sun et al., 2023b).

5.2 Overall Performance

To assess the overall effectiveness of our proposed CORE framework, we first evaluate its performance on the widely-used TREC Deep Learning tracks (DL19–DL22) and a diverse set of BEIR tasks for cross-domain generalization.

Results on TREC Deep Learning Tracks. As shown in Table 1, our CORE-finetuned models demonstrate superior performance over existing state-of-the-art open-source listwise rerankers. Specifically, **CORE_Zephyr** achieves an average nDCG@10 of **0.7517**, surpassing its SFT counterpart RankZephyr (0.7379). Our strongest model, **CORE_Qwen**, further elevates the average performance to **0.7594**, outperforming the highly competitive RankQwen baseline (0.7549). These consistent improvements across most individual tracks highlight CORE's ability to enhance the core ranking effectiveness of LLMs.

Results on BEIR Cross-Domain Tasks. To evaluate generalization capabilities, we test our models on nine diverse tasks from the BEIR benchmark. The results in Table 2 show that **CORE_Qwen** achieves the highest average nDCG@10 score of **0.5632**, outperforming strong baselines like RankQwen (0.5553) and RankZephyr (0.5488). The performance gains are particularly significant on challenging domains such as FiQA and SciFact. This demonstrates that the robust ranking preferences learned through CORE translate well to a wide variety of domains, showcasing its strong generalization ability.

Table 1: Overall Results on TREC DL Tracks (DL19–DL22). Metric is nDCG@10 on top-100 candidates. CORE-finetuned models consistently outperform their SFT counterparts. Best scores are in **bold**, second-best are underlined.

Method	DL19	DL20	DL21	DL22	Average
BM25	0.5058	0.4796	0.4458	0.2692	0.4251
SPLADE++ED	0.7308	0.7195	0.6846	0.5705	0.6764
RankGPT$_4$	0.7464	0.7076	0.7721	0.7175	0.7359
RankVicuna	0.7459	0.7473	0.7011	0.5817	0.6940
RankZephyr	0.7438	0.7620	0.7497	0.6962	0.7379
RankQwen	<u>0.7652</u>	0.7740	<u>0.7534</u>	0.7097	<u>0.7546</u>
CORE_Zephyr	**0.7735**	<u>0.7812</u>	0.7402	<u>0.7120</u>	0.7517
CORE_Qwen	0.7643	**0.8046**	**0.7697**	**0.7231**	**0.7654**

Table 2: Overall Results on BEIR tasks. We report nDCG@10 on the top-100 documents retrieved by Contriever. CORE_Qwen achieves the highest average score, demonstrating strong cross-domain generalization. Best scores are in **bold**, second-best are <u>underlined</u>.

Dataset	Contriever	RankVicuna	RankZephyr	RankQwen	CORE_Qwen
Climate-FEVER	0.237	**0.282**	<u>0.256</u>	0.234	0.223
DBPedia	0.413	0.500	0.500	<u>0.508</u>	**0.512**
FEVER	0.758	0.810	0.801	**0.831**	<u>0.830</u>
FiQA	0.329	0.359	<u>0.422</u>	0.462	**0.478**
HotpotQA	0.638	0.735	0.716	<u>0.740</u>	**0.761**
NFCorpus	0.328	0.331	**0.427**	0.384	<u>0.390</u>
SciDocs	0.165	0.184	**0.377**	0.192	<u>0.208</u>
SciFact	0.677	0.705	0.656	<u>0.768</u>	**0.783**
TREC-COVID	0.596	0.713	0.784	<u>0.879</u>	**0.885**
Average	0.460	0.513	0.549	<u>0.555</u>	**0.563**

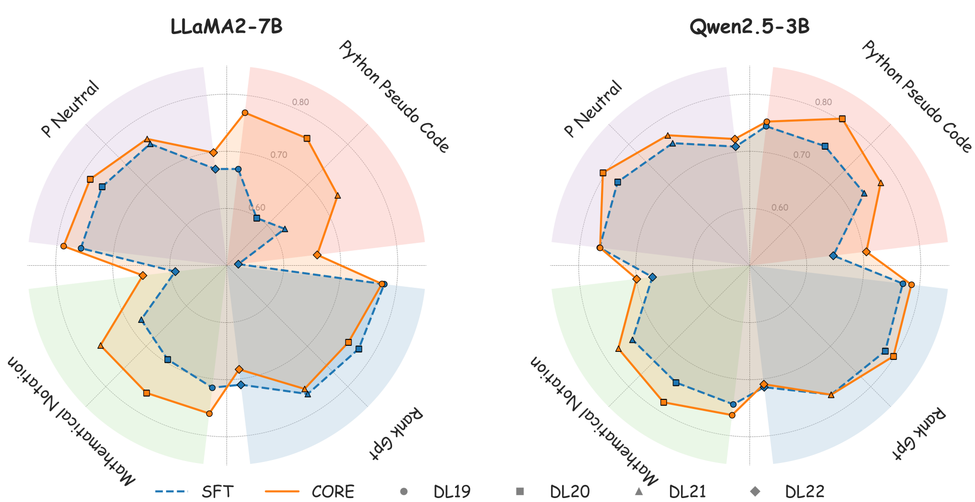

Figure 3: Performance of SFT vs. CORE on LLaMA2-7B and Qwen2.5-3B across four different prompt templates on TREC DL datasets. The radar plots visually demonstrate CORE's key advantage: its performance (solid orange line) is consistently high across all prompts, forming a larger and more regular shape compared to the erratic performance of standard SFT (dashed blue line).

5.3 Robustness to Prompt Variations

The central claim of our work is that CORE can mitigate prompt sensitivity. To verify this, we compare models trained with CORE against standard SFT across four semantically equivalent but stylistically different prompts.

The results, presented visually in Figure 3 and summarized in Table 3, provide strong evidence for our claim. On both LLaMA2-7B and Qwen2.5-3B, the standard SFT model exhibits high performance variance, with scores fluctuating dramatically depending on the prompt. In stark contrast, the CORE-trained models show remarkable stability. For instance, on LLaMA2-7B, CORE reduces the performance spread (max-min difference) from a substantial 0.155 to just 0.028, while also improving the average score. This unequivocally demonstrates that CORE successfully learns a more robust, prompt-invariant ranking function.

Table 3: Summary of prompt robustness on **LLaMA2-7B** and **Qwen2.5-3B**. We report the mean, standard deviation (Std Dev), and performance spread (Max-Min) of nDCG@10 scores across four prompts. CORE significantly reduces variance and improves the average score.

Backbone	Method	Mean	Std Dev	Spread
LLaMA2-7B	SFT	0.6911	0.0681	0.1548
	CORE	**0.7410**	**0.0123**	**0.0275**
Qwen2.5-3B	SFT	0.7385	0.0183	0.0389
	CORE	**0.7611**	**0.0048**	**0.0065**

5.4 Effect of CORE Components

To understand the individual contributions of CORE's key components, we conduct a thorough ablation study across three distinct model backbones of varying sizes: Qwen2.5-0.5B, Qwen2.5-1.5B, and Qwen2.5-3B. The consolidated results are presented in Table 4.

Table 4: Ablation study of CORE components across three different model backbones. We report nDCG@10 on TREC DL datasets. The results show that the full CORE framework consistently achieves the best performance. Best average scores for each backbone are in **bold**.

Backbone	Method	DL19	DL20	DL21	DL22	Avg
Qwen2.5-0.5B	CORE	0.7379	0.7583	0.7219	0.6095	**0.7069**
	w/o $L_{consist}$	0.7313	0.7309	0.6998	0.6088	0.6927
	w/o CFG & $L_{consist}$ (SFT)	0.7308	0.7207	0.6846	0.5705	0.6767
Qwen2.5-1.5B	CORE	0.7590	0.7856	0.7463	0.7084	**0.7498**
	w/o $L_{consist}$	0.7604	0.7860	0.7467	0.6953	0.7471
	w/o CFG & $L_{consist}$ (SFT)	0.7680	0.7642	0.7499	0.7001	0.7456
Qwen2.5-3B	CORE	0.7706	0.7622	0.7705	0.7344	**0.7594**
	w/o $L_{consist}$	0.7693	0.7707	0.7600	0.7282	0.7571
	w/o CFG & $L_{consist}$ (SFT)	0.7652	0.7740	0.7534	0.7097	0.7505

A consistent trend emerges from the results: the full CORE framework consistently yields the best average performance across all model sizes. Removing the internal consistency loss (*w/o* $L_{consist}$) generally leads to a drop in performance, demonstrating the benefit of regularizing the model's internal judgment. A further degradation typically occurs when both the consistency loss and the inverse CFG mechanism are removed, which reduces the model to a standard Supervised Fine-Tuning (SFT) baseline (*w/o CFG &* $L_{consist}$).

Interestingly, the magnitude of the improvement varies with model scale. The impact of the CORE components is most pronounced on the 0.5B model, while the performance differences are more subtle on the 1.5B model. Nevertheless, the full CORE configuration remains the most effective or tied for the best across all tested backbones. These comprehensive results confirm that both the external calibration via Inverse CFG and the internal consistency regularizer are valuable and complementary components for enhancing ranking performance.

6 Conclusion

In this work, we addressed the critical challenge of prompt sensitivity in LLM-based rerankers. We introduced CORE, a novel fine-tuning framework that stabilizes a model's intrinsic ranking preferences. CORE employs a dual strategy, combining an inverse CFG mechanism for external behavior calibration with a differentiable Kendall's Tau loss for internal judgment consistency. Experiments on TREC DL and BEIR benchmarks confirm that CORE achieves state-of-the-art performance and yields significantly more robust and stable rankings across diverse prompts. Our work represents a key step towards more reliable LLM systems, and the proposed framework offers a promising direction for mitigating input sensitivity in other text generation tasks.

REFERENCES

Negar Arabzadeh and Charles L. A. Clarke. A human-ai comparative analysis of prompt sensitivity in llm-based relevance judgment. *CoRR*, abs/2504.12408, 2025.

Anwoy Chatterjee, H. S. V. N. S. Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. POSIX: A prompt sensitivity index for large language models. In *EMNLP Findings*, pp. 14550–14565, 2024.

Yiqun Chen, Qi Liu, Yi Zhang, Weiwei Sun, Xinyu Ma, Wei Yang, Daiting Shi, Jiaxin Mao, and Dawei Yin. Tourrank: Utilizing large language models for documents ranking with a tournament-inspired strategy. In *WWW*, 2025.

Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. CFG++: manifold-constrained classifier free guidance for diffusion models. In *ICLR*, 2025.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the TREC 2019 deep learning track. *CoRR*, abs/2003.07820, 2020.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the TREC 2020 deep learning track. *CoRR*, abs/2102.07662, 2021.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. Overview of the TREC 2021 deep learning track. *CoRR*, abs/2507.08191, 2025a.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. Overview of the TREC 2022 deep learning track. *CoRR*, abs/2507.10865, 2025b.

Yongqi Fan, Xiaoyang Chen, Dezhi Ye, Jie Liu, Haijin Liang, Jin Ma, Ben He, Yingfei Sun, and Tong Ruan. Tfrank: Think-free reasoning enables practical pointwise LLM ranking. *CoRR*, abs/2508.09539, 2025.

Vincent Fortuin. Priors in bayesian deep learning: A review. *CoRR*, abs/2105.06868, 2021.

Sigmund Freud. *The Ego and the Id*. Internationaler Psychoanalytischer Verlag, 1923.

Jingtong Gao, Bo Chen, Xiangyu Zhao, Weiwen Liu, Xiangyang Li, Yichao Wang, Wanyu Wang, Huifeng Guo, and Ruiming Tang. Llm4rerank: Llm-based auto-reranking framework for recommendations. In *WWW*, pp. 228–239, 2025.

Yuchen Guan, Runxi Cheng, Kang Liu, and Chun Yuan. Kendall's τ coefficient for logits distillation. *CoRR*, abs/2409.17823, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598, 2022.

Chi Hu, Yuan Ge, Xiangnan Ma, Hang Cao, Qiang Li, Yonghua Yang, Tong Xiao, and Jingbo Zhu. Rankprompt: Step-by-step comparisons make language models better reasoners. In *COLING*, pp. 13524–13536, 2024.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. Leveraging passage embeddings for efficient listwise reranking with large language models. In *WWW*, pp. 4274–4283, 2025.

Kehan Long, Shasha Li, Chen Xu, Jintao Tang, and Ting Wang. Precise zero-shot pointwise ranking with llms through post-aggregated global context information. *CoRR*, abs/2506.10859, 2025.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*, 2023.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In *NIPS*, 2016.

Lilian Ngweta, Kiran Kate, Jason Tsay, and Yara Rizk. Towards llms robustness to changes in prompt format styles. In *NAACL*, 2025.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *CoRR*, abs/2312.02724, 2023a.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *CoRR*, abs/2309.15088, 2023b.

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. Prompt perturbation consistency learning for robust language models. In *EACL Findings*, pp. 1357–1370, 2024.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In *NAACL Findings*, pp. 1504–1518, 2024.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md. Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. FIRST: faster improved listwise reranking with single token decoding. In *EMNLP*, 2024.

Ruiyang Ren, Yuhao Wang, Kun Zhou, Wayne Xin Zhao, Wenjie Wang, Jing Liu, Ji-Rong Wen, and Tat-Seng Chua. Self-calibrated listwise reranking with large language models. In *WWW*, pp. 3692–3701, 2025.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. Improving passage retrieval with zero-shot question generation. In *EMNLP*, 2022.

Dylan Sam, Rattana Pukdee, Daniel P. Jeong, Yewon Byun, and J. Zico Kolter. Bayesian neural networks with domain knowledge priors. *CoRR*, abs/2402.13410, 2024.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *ICLR*, 2024a.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *ICLR*, 2024b.

Shuoqi Sun, Shengyao Zhuang, Shuai Wang, and Guido Zuccon. An investigation of prompt variations for zero-shot llm-based rankers. In *ECIR*, volume 15573, pp. 185–201, 2025.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *EMNLP*, pp. 14918–14937, 2023a.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *EMNLP*, pp. 14918–14937, 2023b.

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. In *NAACL*, pp. 2327–2340, 2024.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS*, 2021.

Jim Waldo and Soline Boussard. Gpts and hallucination: Why do large language models hallucinate? *ACM Queue*, 22(4):10, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

Chenxing Wei, Yao Shu, Mingwen Ou, Ying Tiffany He, and Fei Richard Yu. PAFT: prompt-agnostic fine-tuning. *CoRR*, abs/2502.12859, 2025.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang (eds.), *ICML*, pp. 12697–12706, 2021.

Kaipeng Zheng, Huishuai Zhang, and Weiran Huang. Diffkendall: A novel approach for few-shot learning with differentiable kendall's rank correlation. In *NeurIPS*, 2023.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A. Heller, and Subhrajit Roy. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. In *ICLR*, 2024.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning T5 for text ranking with ranking losses. In *SIGIR*, pp. 2308–2313, 2023.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. In *NAACL*, 2024a.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *SIGIR*, 2024b.

A Appendix

A.1 Implementation Details

Training. All models were fine-tuned on the full dataset provided by RankZephyr (Pradeep et al., 2023a), which comprises 39,912 training instances. Our experiments were conducted on a single NVIDIA A40 GPU with 48GB of VRAM. To manage memory and facilitate training of larger models, we utilized the DeepSpeed framework with the ZeRO Stage 3 optimization and CPU offloading enabled. Key hyperparameters were kept consistent across all experiments to ensure a fair comparison. We used the AdamW optimizer with a learning rate of **5e-6**, scheduled using a cosine decay with **50** warmup steps. We used a per-device batch size of **2** and a gradient accumulation of **16**, resulting in an effective batch size of 32. For reproducibility, the global random seed was set to **42** for all runs.

Inference. Our inference process follows a standard two-stage retrieve-and-rerank pipeline. For all experiments on the TREC DL and BEIR benchmarks, we first use the SPLADE++ (EnsembleDistil ONNX version) retriever to generate a candidate pool of the top 100 documents for each query. In the second stage, our LLM reranker processes these 100 documents using a sliding window approach, following the methodology of RankZephyr (Pradeep et al., 2023a). We use a window size of 20 and a stride of 10, requiring 9 slides to cover the full list. After aggregating the results from all windows, the final output consists of the top 20 reranked documents for evaluation.

A.2 Prompt Templates

Our robustness experiments utilized one **neutral prompt** ($p_{neutral}$) and three **diverse prompts** ($\mathcal{P}_{train}$). The neutral prompt is a simple, direct instruction for the ranking task. The diverse prompts are designed to be semantically equivalent but stylistically different, framing the task as a general AI assistant instruction (RankGPT style), a piece of Python pseudocode, and a mathematical notation problem, respectively.

Below are the system messages and user-facing prompt templates used in our experiments. The "query" and "documents" placeholders are dynamically filled during runtime.

Neutral Prompt This is the standard, task-focused prompt used for all main evaluations and as the baseline for training CORE.

```
# System Message
You are an AI assistant tasked with ranking documents based on relevance
    ↪ to a query.
Your response must be a direct sequence of alphabetical document IDs,
    ↪ ordered from
most to least relevant, in the format [A] > [B] > ... > [N]. Provide
    ↪ nothing else.

# User Prompt Template
Rank the following {document_num} passages, identified by alphabetical
    ↪ IDs [],
based on their relevance to the query: {query}.

Query: {query}

Documents:
{documents}

Your output must be a ranked list of the alphabetical passage IDs, in
    ↪ descending
order of relevance, formatted strictly as: [A] > [B] > ... > [N].
Provide only this ranked list.
% \end{verbatim}
```

Diverse Prompt 1: RankGPT Style This prompt mimics the conversational style of a general-purpose AI assistant.

```
# System Message
You are RankLLM, an intelligent assistant that can rank passages based
    ↪ on their
relevancy to the query.

# User Prompt Template
I will provide you with {label_num} passages, each indicated by a
    ↪ alphabetic
identifier []. Rank the passages based on their relevance to the search
    ↪ query: {query}

{documents}

Search Query: {query}

Rank the {label_num} passages above based on their relevance to the
    ↪ search query.
All the passages should be included and listed using identifiers, in
    ↪ descending order
of relevance. The output format should be [] > [], e.g., [A] > [B]. Only
    ↪ respond
with the ranking results, do not say any word or explain.
% \end{verbatim}
```

Diverse Prompt 2: Python Pseudocode This prompt frames the task as the execution of a Python function, testing the model's ability to follow structured, code-like instructions.

```
# System Message
You are an AI engine that interprets pseudocode defining a ranking task.
    ↪ Your output
```

```
must be the result of the described ranking function, formatted as a
    ↪ string:
[A] > [B] > ... > [N], representing alphabetically identified documents
    ↪ in
descending order of relevance. Output only this string.

# User Prompt Template
# Function Definition: PerformRelevanceRanking
# Objective: Order a list of documents based on their relevance to a
    ↪ given query.

def perform_relevance_ranking(query_text: str, input_document_data: str):
    """
    Ranks documents provided in 'input_document_data' against the
        ↪ 'query_text'.
    The 'input_document_data' is a string containing documents, each
        ↪ with an
    alphabetical ID. The alphabetical IDs from the input should be used
        ↪ in the output.
    """
    current_query = "{query}"
    candidate_documents_text_block = """{documents}"""

    # --- Ranking Logic (To Be Performed by You) ---
    # Your goal is to determine the 'relevant_order'' based on
        ↪ 'current_query'
    # and the information within 'candidate_documents_text_block'.
    # ------------------------------------------------

    # Output Specification:
    # Return a string representing the sorted alphabetical document IDs,
    # from most relevant to least relevant.
    # Format: "[A] > [B] > ... > [N]"
    pass # Replace with actual output string
% \end{verbatim}
```

Diverse Prompt 3: Mathematical Notation This prompt presents the task in a formal, mathematical style, testing the model's ability to parse symbolic instructions.

```
# System Message
You are an AI system designed to interpret ranking tasks defined with
mathematical-like notation. Your role is to compute the ranking R*. The
    ↪ output must
be a string of alphabetical document IDs: [A] > [B] > ... > [N], ordered
    ↪ by a
relevance function decreasingly. Provide only this string.

# User Prompt Template
Let Q be the query:
Q = "{query}"

Let D be the set of {document_num} documents, D = {d_A, d_B, ..., d_N}.
Each document d_i has a unique alphabetical identifier ID(d_i).
The document content is provided in {documents}.

Define a relevance function, Rel(Q, d_i), which scores the relevance of
document d_i to query Q.

The task is to find an ordered sequence of alphabetical document
    ↪ identifiers R*:
R* = [ID(d_j1)] > [ID(d_j2)] > ... > [ID(d_jN)]
such that Rel(Q, d_j1) >= Rel(Q, d_j2) >= ... >= Rel(Q, d_jN).
```

```
Provide the sequence R* as a string:
% \end{verbatim}
```

A.3 Statement on AI Usage

In preparing this manuscript, we leveraged large language models to improve academic writing and to assist in debugging code. These tools served a function analogous to that of a human copyeditor or a programming linter, with their use solely dedicated to enhancing clarity, grammatical correctness, and code efficiency. All conceptual insights, methodological designs, experimental results, and critical analyses presented in this work remain the original contributions of the authors.