PLR: Plackett–Luce for Reordering In-Context Learning Examples

Anonymous ACL submission

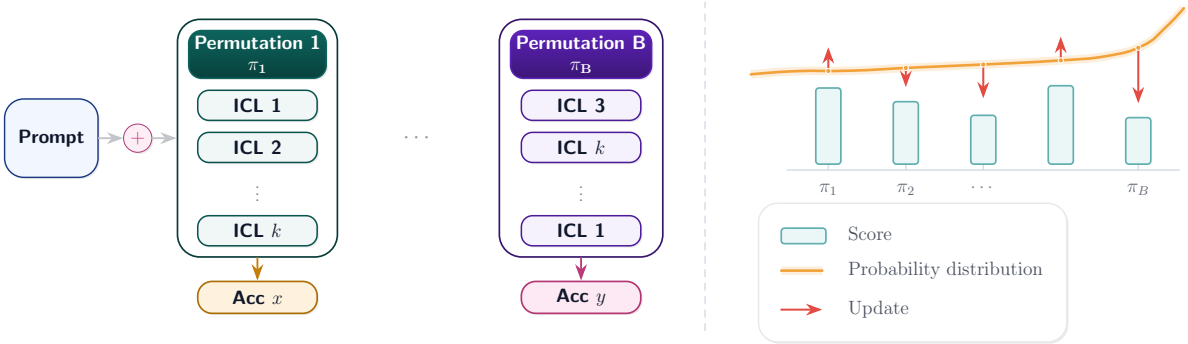

Figure 1: Overview of our probabilistic approach to in-context example ordering. We maintain a Plackett-Luce distribution over permutations, repeatedly sample ICL example orderings, evaluate them under a task-level metric, and update the distribution to shift probability mass toward high-performing orders and away from low-performing ones.

Abstract

In-context learning (ICL) adapts large language models by conditioning on a small set of ICL examples, avoiding costly parameter updates. Among other factors, performance is often highly sensitive to the *ordering* of the examples. However, exhaustive search over the $n!$ possible orderings is infeasible. Therefore more efficient ordering methods use model confidence measures (e.g., label-probability entropy) over label sets or take a direct approach to finding the best ordering. We propose PLR, a probabilistic approach to in-context example ordering that replaces discrete ordering search with learning a probability distribution over orderings with the Plackett–Luce model. PLR models orderings using a Plackett–Luce distribution and iteratively updates its parameters to concentrate probability mass on high-performing orderings under a task-level metric. Candidate orderings are sampled efficiently via a Gumbel perturb-and-sort procedure. Experiments on multiple classification benchmarks show that PLR consistently improves few-shot accuracy for $k \in \{4, 8, 16, 32\}$ examples, and we further demonstrate gains on mathematical reasoning tasks where label-based ordering methods are not applicable.

1 Introduction

Prompting LLMs with in-context examples is a practical alternative to fine-tuning for adapting LLMs (Sahoo et al., 2024; Chen et al., 2023a). While its mechanisms remain under active study, recent analyses suggest transformers can treat examples as a small inference-time "training set," implicitly adapting from context (Akyürek et al., 2022; Von Oswald et al., 2023; Xie et al., 2022). However, ICL can be brittle, minor semantics-preserving prompt edits often cause large performance changes (Razavi et al., 2025; Sclar et al., 2024; Chatterjee et al., 2024; Zhuo et al., 2024; Errica et al., 2024; Cao et al., 2024), including under natural perturbations and adversarially injected instructions (Zhu et al., 2024; Li et al., 2023). Another surprising aspect is that performance depends not only on which demonstrations are used but also on their order (Zhao et al., 2021a; Bhope et al., 2025; He et al.; Lu et al., 2022a; Min et al., 2022a; Reynolds and McDonell, 2021). Similar sensitivity appears in newer reasoning-oriented models (Guo et al., 2025).

The space of possible orders grows as $n!$, making exhaustive search for finding an optimal ordering infeasible. Existing methods therefore rely on

heuristics or limited candidate evaluation, including entropy- and label-distribution–based criteria such as LocalE/GlobalE (Lu et al., 2022a) and PDO (Xu et al., 2024) and dataset-free filtering schemes like DEmO (Guo et al., 2024). These approaches work well for classification but typically require enumerating a finite label set (via verbalizers), which does not extend to open-ended generation and numerical reasoning with unbounded outputs (e.g., GSM8K). They also often perform brittle point-estimate in which a single ordering is sought, and may require factorial-time enumeration in small-n regimes (Bhope et al., 2025). To address these limitations, we propose PLR, which replaces discrete permutation search with learning a distribution over orderings. By modeling the space of orderings with a Plackett–Luce (PL) distribution (and mixtures thereof) and updating it to concentrate mass on high-performing regions, PLR robustly and efficiently explores the permutation space and directly optimizes the task metric without assuming a finite label space.

To summarize, our contributions are as follows:

Conceptual: We introduce PLR, a distributional approach to in-context example ordering that learns to place higher probability on better-performing example orderings under a task-level score. Unlike prior label-probability heuristics that require enumerating a finite label set, PLR is label-space agnostic and applies naturally to open-ended generation and numerical reasoning. We use the (mixture of) Plackett–Luce distribution and use Gumbel perturb-and-sort to sample efficiently from it.

Algorithmic: We propose an iterative algorithm involving sampling for fitting the PL distribution, we propose three approaches: (i) a heuristic rank update, (ii) MLE and (iii) EM for the mixture of PL. We stabilize updates with EMA.

Experimental: We evaluate PLR on classification and reasoning benchmarks with Qwen and Llama models, and provide ablations analyzing key design choices. Our experiments show significantly higher performance than our baseline methods.

2 Related Work

Prompt Engineering and In-Context Prompt Optimization. Prompt engineering can substantially improve LLM performance without parameter updates (Liu et al., 2023), including reasoning-oriented methods such as Chain-of-Thought and self-consistency (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022) and more structured variants (e.g., Tree-/Program-/Graph-of-Thought and decomposition) (Yao et al., 2023; Chen et al., 2022; Besta et al., 2024; Zhou et al., 2023). Beyond manual design, automated prompt optimization searches over instructions and templates via model-guided refinement, evolutionary strategies, or black-box optimization (Zhou et al., 2022; Pryzant et al., 2023; Guo et al., 2023; Fernando et al., 2023; Batorski and Swoboda, 2025a; Wang et al., 2023b; Yang et al., 2024b), and related work optimizes discrete prompt tokens or uses strong random-search baselines (Shin et al., 2020; Lu et al., 2024b). For in-context learning, performance also depends on which demonstrations are included and their *ordering* (Brown et al., 2020; Lu et al., 2022b), yet most prompt optimizers emphasize instruction text, treat demonstrations in a restricted way, or incur high cost when generating new examples (Deng et al., 2022; Batorski et al., 2025; Agarwal et al., 2024; Batorski and Swoboda, 2025b). We target this underexplored axis by optimizing the ordering of a fixed set of in-context examples.

ICL examples sensitivity. In-context learning can be fragile: even with the task and model fixed, small changes to a prompt with ICL examples e.g., which examples are used, their formatting/label strings, or their position in the context can cause large performance swings (Brown et al., 2020; Zhao et al., 2021b; Lu et al., 2024a; Zhang et al., 2024). This sensitivity is often attributed to models relying on shallow patterns in examples (e.g., label/format heuristics) rather than task semantics (Min et al., 2022b; Fei et al., 2023; Jang et al., 2024; Sclar et al., 2023; Chen et al., 2023b; Liang et al., 2025; Fang et al., 2025). Positional effects further amplify brittleness, with systematic biases toward certain prompt regions (Cobbina and Zhou, 2025; Guo and Vosoughi, 2025; Xiang et al., 2024) and reduced use of evidence in long contexts, especially for information in the middle (Liu et al., 2024). These findings motivate methods that explicitly optimize and robustify prompt construction, including ICL example ordering (Lu et al., 2024a; Zhang et al., 2024).

ICL Example Ordering Methods. A growing line of work studies how to order a fixed set of in-

context examples to reduce brittleness. Prior analyses report strong positional and recency effects, where the same examples can perform very differently depending on their arrangement (Shin et al., 2022; Min et al., 2022b; Wu et al., 2024; Rubin et al., 2022). Accordingly, many methods score or search over a subset of candidate orderings using proxy signals e.g., confidence/entropy heuristics, label-distribution matching, or other task-specific criteria to avoid enumerating the full $n!$ space (Lu et al., 2022b; Xu et al., 2024; Guo et al., 2024; Wang et al., 2023a; Yang et al., 2023). More structured procedures, such as optimization-based search and rapid candidate refinement, can further improve reliability but still evaluate only a limited set of orders (Pham et al., 2025; He et al.). In contrast, we perform distributional optimization over orderings to directly optimize the task metric. This allows us to frame the optimization in a principled manner and, on a practical side, to explore efficiently the large underlying search space.

3 Method

Problem Statement. Let $E = \{(x_i, y_i)\}_{i=1}^n$ be a fixed set of n in-context examples, where x_i is an input/instruction and y_i is its corresponding answer (label). Let S_n denote the symmetric group on n elements, i.e., the set of all $n!$ orderings (or, mathematically, permutations) of $\{1, \ldots, n\}$. For any permutation $\pi = (\pi_1, \ldots, \pi_n) \in S_n$, define the ordered example sequence

$$E_\pi \;=\; \big((x_{\pi_1}, y_{\pi_1}), \ldots, (x_{\pi_n}, y_{\pi_n})\big).$$

Let p be a fixed user-provided instruction (prefix prompt), and let $\oplus$ denote prompt concatenation. The full prompt induced by π is

$$P(\pi) \;=\; p \oplus E_\pi.$$

Given an evaluation dataset D and a scoring function f (e.g., accuracy) that evaluates the model on D using prompt $P(\pi)$, our goal is to find

$$\pi^\star \;=\; \arg\max_{\pi \in S_n} \; f\big(D,\; p \oplus E_\pi\big).$$

3.1 Model

We will formulate finding the optimal ordering as fitting a distribution over permutations such that high quality permutations are given high probability mass. To this end, we propose to use the Plackett–Luce (PL) distribution due to its computational efficiency and extend its expressive power by considering the mixture of PLs.

Framing example ordering as distribution estimation. Rather than searching directly for a single optimal ordering $\pi^\star \in S_n$, we maintain a parametric distribution $q_\phi(\pi)$ over permutations and optimize its parameters ϕ so that high-probability permutations achieve high task performance. Concretely, given an evaluation dataset D, a fixed prompt prefix p, and a scoring function $f(D, p \oplus E_\pi)$, we aim to maximize the expected score under q_ϕ:

$$\phi^\star \;=\; \arg\max_{\phi} \; E_{\pi \sim q_\phi}\Big[f\big(D,\; p \oplus E_\pi\big)\Big]. \qquad (1)$$

In our setting, q_ϕ is instantiated as a Plackett–Luce distribution, which provides an efficient mechanism to sample candidate permutations and progressively concentrate probability mass on better-performing orderings.

Plackett–Luce distribution. The Plackett–Luce (PL) model (Luce, 1959; Plackett, 1975) defines a probability distribution over permutations $\pi = (\pi_1, \ldots, \pi_n) \in S_n$ parameterized by a score (logit) vector $\theta \in \mathbb{R}^n$. The model can be interpreted as a sequential choice process: at rank position r, the item placed at that position is drawn from the set of remaining items with probability proportional to its exponentiated score. Formally,

$$\Pr(\pi \mid \theta) = \prod_{r=1}^{n} \frac{\exp(\theta_{\pi_r})}{\sum\limits_{j \in R_r} \exp(\theta_j)}, \qquad (2)$$

where $R_r = \{\pi_r, \pi_{r+1}, \ldots, \pi_n\}$ denotes the set of items not yet selected at step r. The PL distribution is invariant to additive shifts of the logits, i.e., $\Pr(\pi \mid \theta) = \Pr(\pi \mid \theta + c\mathbf{1})$ for any $c \in \mathbb{R}$, and it is therefore common to impose an identifiability constraint such as $\sum_{i=1}^n \theta_i = 0$.

Mixture of PL distributions. A single PL model uses one global logit vector θ to generate a ranking via sequential choices. Although the selection is conditioned on previously chosen items, this conditioning operates only through removal: the same logits are reused at every step on the remaining set. In particular, for any two items a, b that are still available at position r, PL satisfies an independence-of-irrelevant-alternatives (IIA) property (Luce, 1959; Plackett, 1975),

$$\frac{\Pr(\pi_r = a \mid \pi_{<r}; \theta)}{\Pr(\pi_r = b \mid \pi_{<r}; \theta)} = \exp(\theta_a - \theta_b),$$

which is invariant to the previously selected items $\pi_{<r}$ (except that selected items are excluded). Consequently, a single PL component cannot express interaction effects such as "if example i is placed first, then example j should be placed second" beyond the trivial fact that i is no longer eligible. To partially relax this limitation while retaining efficient sampling, we consider a mixture of PL distributions (Zhao et al., 2016; Zhao and Xia, 2019):

$$q(\pi) \;=\; \sum_{k=1}^{K} \alpha_k \, \mathrm{PL}\Big(\pi \mid \theta^{(k)}\Big)\,,$$

where $\alpha_k \geq 0, \quad \sum_{k=1}^{K} \alpha_k = 1$. The mixture introduces a latent variable $z \in \{1, \ldots, K\}$ that indicates from which mixture component the permutation was drawn. The mixture of PL distribution allows to model distributions that are not subject to the IIA condition, even though each individual mixture distribution does so. This increases expressivity for multi-modal sets of high-performing demonstration orders while preserving the same perturb-and-sort sampling mechanism within each component.

Even stronger, it is known that mixture of PL can approximate any distribution over permutations arbitrarily well when the number of mixture components K is large enough, see Theorem 1 in the appendix.

Gumbel trick for sampling PL permutations. Sampling from a Plackett–Luce model can be done via sequential softmax choices over the remaining items, but this procedure is inherently sequential and can be inefficient when drawing many permutations, therefore we use the Gumbel Trick (Kool et al., 2019; Maddison et al., 2014) which simplifies the sampling problem to sampling from a Gumbel distribution. Let $\theta \in R^n$ be the PL logits, and draw i.i.d. Gumbel noise variables

$$g_i \sim \mathrm{Gumbel}(0, 1), \qquad i = 1, \ldots, n,$$

which can be obtained from $u_i \sim \mathrm{Uniform}(0, 1)$ by

$$g_i = -\log\big(-\log(u_i)\big).$$

Define perturbed scores $s_i = \theta_i + g_i$ and sort items by decreasing s_i. Denoting by π the resulting ordering (i.e., $s_{\pi_1} \geq s_{\pi_2} \geq \cdots \geq s_{\pi_n}$), the induced random permutation satisfies

$$\pi \sim \mathrm{PL}(\theta).$$

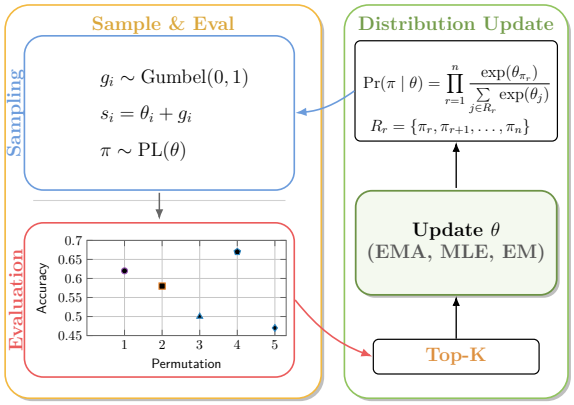

Figure 2: Illustration of PLR. Given a Plackett–Luce distribution, we sample high-probability permutations with the Gumbel trick. Each permutation is scored and the top-K are retained and used for fitting an improved Plackett-Luce distribution. This is iterated until high quality ICL ordering is found.

Intuitively, the Gumbel perturbations transform sampling from a sequence of softmax choices into a single "perturb-and-sort" operation, enabling efficient and numerically stable generation of full rankings from the PL model.

For the mixture of PL we first sample the mixture component using the mixture weights α_i, after which we sample as above from the selected PL distribution.

3.2 Training

We illustrate our approach to finding an ordering via fitting a PL distribution in Algorithm 1. We start with a uniform initial distribution (line 2), sample a number of permutations (lines 4-10) and evaluate their performance on the training set (line 11), fit the distribution to the best permutations according to the training set (line 13) and iterate. The final permutation is obtained by sampling several permutations from the obtained distribution (line 16) and taking the best according to a held-out validation set (line 18). The extension to mixture of PL is given straightforwardly by replacing all PL specific steps by the mixture of PL counterparts. A graphical illustration of the overall procedure is given in Figure 2.

We propose three ways to fit the (mixture of) PL distributions below.

Training the Plackett–Luce parameters. We propose three ways to estimate PL parameters: a heuristic rank update and using MLE for the PL distribution and EM for the mixture of PL. We sta-

Algorithm 1 PLR

Require: ICL examples $E = \{(x_i, y_i)\}_{i=1}^{n}$, prefix prompt p, metric $f(\cdot)$, model $\mathcal{M}$
Require: training sets $\{D_t^{\text{train}}\}_{t=1}^{T}$, validation set D^{val}
Require: CE params: batch B, elite fraction ρ, final draws K
Require: update routine $\textsc{Update}(\theta; \mathcal{E}_t, \{s^{(b)}\}_{\pi^{(b)} \in \mathcal{E}_t}, \ldots)$
1: $\theta \leftarrow \mathbf{0} \in R^n$ ▷ PL logits: enforce $\sum_i \theta_i = 0$
2: **for** $t = 1, \ldots, T$ **do**
Sample B **permutations via Gumbel perturb-and-sort:**
3: **for** $b = 1, \ldots, B$ **do**
4: **for** $i = 1, \ldots, n$ **do**
5: Sample $u_i \sim \text{Uniform}(0, 1)$
6: $g_i \leftarrow -\log\big(-\log(u_i)\big)$ ▷ $g_i \sim \text{Gumbel}(0, 1)$
7: $\tilde{s}_i \leftarrow \theta_i + g_i$
8: **end for**
9: $\pi^{(b)} \leftarrow \text{argsort}_{i \in \{1,\ldots,n\}}\big(\tilde{s}_i\big)$ (descending)
10: **end for**
11: Score $s^{(b)} \leftarrow f\big(D_t^{\text{train}},\ p \oplus E_{\pi^{(b)}};\ \mathcal{M}\big)$ for all b
12: Let $\mathcal{E}_t \leftarrow \textsc{Top}_{\lceil \rho B \rceil}\big(\{\pi^{(b)}\}, \{s^{(b)}\}\big)$
13: $\theta \leftarrow \textsc{Update}\Big(\theta;\ \mathcal{E}_t,\ \{s^{(b)}\}_{\pi^{(b)} \in \mathcal{E}_t}, \ldots\Big)$
14: $\theta \leftarrow \theta - \frac{1}{n}\sum_{i=1}^{n} \theta_i$ ▷ center logits
15: **end for**
16: Draw $\{\pi^{(k)}\}_{k=1}^{K} \sim \text{PL}(\cdot \mid \theta)$ (same Gumbel trick) and select
17: $\hat{\pi} \leftarrow \arg\max_{\pi^{(k)}} f\big(D^{\text{val}},\ p \oplus E_{\pi^{(k)}};\ \mathcal{M}\big)$
18: **return** $\hat{\pi}$

bilize all by using exponential moving averages.

The learning happens on line 13 of Algorithm 1. We are given the current PL logits $\theta \in R^n$ (and, for mixtures, weights) and a set of high-scoring (*elite*) permutations produced during the optimization procedure $\mathcal{E} = \{\pi^{(1)}, \ldots, \pi^{(M)}\} \subset S_n$, optionally with nonnegative weights $\{w_m\}_{m=1}^{M}$ (e.g., proportional to training accuracy).

(i) Exponential moving average (EMA) / heuristic rank update. A simple update constructs a target score vector from the average elite ranks. Let $\text{rank}_\pi(i) \in \{0, \ldots, n-1\}$ be the position of item i in permutation π (higher logits correspond to earlier position in the permutation). Define the elite average rank

$$\bar{r}_i \;=\; \frac{1}{M}\sum_{m=1}^{M} \text{rank}_{\pi^{(m)}}(i),$$

and a rank-to-logit target (with temperature $\tau > 0$)

$$\theta_i^{\text{tgt}} \;=\; -\frac{\bar{r}_i}{\tau}.$$

Thus, if the average elite rank is low (i.e. example occurs early), then its logits should be increased in comparison to other logits (corresponding to small $\bar{r}_i/\tau$). Vice versa for high rank.

We then apply an EMA-style update with step size $\alpha \in [0, 1]$:

$$\theta \;\leftarrow\; (1-\alpha)\,\theta \;+\; \alpha\,\theta^{\text{tgt}}.$$

Finally, we enforce identifiability and numerical stability by centering and clipping, e.g. $\theta \leftarrow \theta - \frac{1}{n}\sum_{i=1}^{n}\theta_i$ and $\theta_i \in [-c, c]$.

(ii) Maximum likelihood (MLE) on elite permutations. A more principled alternative fits θ by maximizing the (weighted) PL log-likelihood of the elite permutations:

$$\theta^{\text{MLE}} \;\in\; \arg\max_{\theta \in R^n} \sum_{m=1}^{M} w_m \log \Pr(\pi^{(m)} \mid \theta),$$

where $\Pr(\pi \mid \theta)$ is given in Eq. (2). In practice we solve this optimization with gradient-based methods (e.g. Adam), again applying a centering constraint to resolve the additive invariance of θ. To stabilize optimization across iterations, we optionally combine MLE with an EMA step:

$$\theta \;\leftarrow\; (1-\alpha)\,\theta \;+\; \alpha\,\theta^{\text{MLE}}.$$

The MLE is well-behaved due to the Plackett–Luce log-likelihood being concave in θ (Maystre and Grossglauser, 2015).

EM-style training for the mixture. We fit the mixture parameters $(\alpha, \{\theta^{(k)}\}_{k=1}^{K})$ using an EM-style procedure on the current elite set of permutations $\mathcal{E} = \{\pi^{(m)}\}_{m=1}^{M}$, optionally weighted by w_m (e.g., proportional to training accuracy). In the E-step, we compute responsibilities

$$r_{mk} = \frac{\alpha_k \,\text{PL}(\pi^{(m)} \mid \theta^{(k)})}{\sum_{\ell=1}^{K} \alpha_\ell \,\text{PL}(\pi^{(m)} \mid \theta^{(\ell)})}.$$

In the M-step, we update mixture weights via $\alpha_k \leftarrow \frac{\sum_m w_m r_{mk}}{\sum_m w_m}$ and update each component logits by weighted maximum likelihood,

$$\theta^{(k)} \leftarrow \arg\max_{\theta} \sum_{m=1}^{M} w_m r_{mk} \log \text{PL}(\pi^{(m)} \mid \theta),$$

which we optimize approximately with a few steps of Adam and then optionally smooth with an EMA update for stability. This EM-style update allows the mixture to allocate different components to different modes of high-performing orderings while preserving efficient PL sampling within each component.

Table 1: Test accuracy (%) averaged over seeds. best score is colored in red, second best in orange and third best in yellow. Avg is macro-average over MR/NEWS/SST5/TREC/SUBJ. Top block: `qwen2.5-7b-instruct`. Bottom block: `Llama3.1-8B-instruct`.

Method	Model	MR				NEWS				SST5				TREC				SUBJ				Avg			
		k=4	k=8	k=16	k=32	k=4	k=8	k=16	k=32	k=4	k=8	k=16	k=32	k=4	k=8	k=16	k=32	k=4	k=8	k=16	k=32	k=4	k=8	k=16	k=32
Static	Qwen2.5-7B-Instruct	90.22	91.10	91.75	91.73	83.37	82.47	83.71	83.79	54.78	54.12	54.78	56.52	61.95	63.90	60.26	59.99	65.88	71.50	76.55	79.23	71.24	72.62	73.41	74.25
Top-K		90.90	91.14	91.95	91.96	84.18	84.40	85.07	85.35	55.55	55.36	55.85	56.85	65.79	65.45	61.00	63.29	71.54	77.43	79.69	82.32	73.59	74.76	74.71	75.95
LocalE		90.72	91.20	91.36	91.90	82.59	83.53	83.69	82.76	54.72	54.55	54.67	56.86	63.87	65.86	61.14	61.74	69.07	73.70	75.96	80.05	72.19	73.77	73.36	74.66
GlobalE		90.66	91.11	91.69	91.80	82.79	83.60	83.81	82.97	54.50	54.47	55.07	56.53	59.31	63.02	61.67	61.07	68.48	70.93	74.64	77.82	71.15	72.63	73.38	74.04
PDO		90.56	91.08	91.39	91.73	84.11	83.50	82.64	84.23	54.87	54.34	55.31	56.29	59.01	64.10	62.21	61.81	65.69	70.12	75.26	77.00	70.85	72.63	73.36	74.21
PDO-U		90.61	90.91	91.87	91.85	83.94	84.06	84.73	85.13	54.64	55.51	55.28	57.32	60.33	64.03	63.49	63.16	72.83	76.79	78.82	81.23	72.47	74.26	74.84	75.74
PDO-Up		90.61	91.10	91.87	91.83	83.79	83.98	84.73	85.34	54.76	55.60	55.76	57.28	59.32	65.32	63.22	62.08	72.83	76.79	78.82	81.23	72.26	74.56	74.88	75.55
DeMO		90.79	91.17	91.67	91.70	83.50	83.91	83.66	84.77	55.37	54.47	55.66	57.06	63.06	66.13	63.97	65.59	68.95	72.06	74.15	78.94	72.33	73.55	73.82	75.61
PLR-EMA		90.67	91.46	91.83	92.21	84.30	84.80	86.14	86.38	55.38	55.41	56.74	57.18	65.29	66.60	62.08	62.15	72.78	80.64	83.98	86.75	73.68	75.78	76.15	76.93
PLR-1		90.78	91.32	91.95	92.14	84.34	84.89	85.80	86.82	55.08	55.55	56.78	56.86	66.91	65.99	61.47	61.20	72.80	80.33	83.44	85.90	73.98	75.62	75.89	76.58
PLR-4		90.82	91.40	91.79	92.02	84.29	84.97	86.28	85.98	55.58	56.04	57.18	57.81	66.60	66.94	62.28	62.08	73.09	80.94	85.17	85.98	74.08	76.06	76.54	76.77
Static	Llama3.1-8B-Instruct	91.34	91.87	92.07	92.39	79.18	80.99	82.59	83.02	50.14	49.50	51.93	53.57	53.12	58.95	56.80	65.28	79.76	84.35	84.12	87.17	70.71	73.13	73.50	76.29
Top-K		91.49	91.99	92.44	93.11	81.82	83.66	85.00	85.30	51.27	53.91	54.84	54.33	56.38	63.35	62.34	66.25	85.61	86.44	90.12	91.15	73.31	75.87	76.95	78.03
LocalE		91.53	91.95	91.87	92.71	80.11	81.35	83.27	82.72	50.48	51.17	52.50	54.08	58.01	60.96	58.01	58.15	81.41	81.04	82.37	87.54	72.31	73.29	73.60	75.04
GlobalE		91.42	91.90	91.92	92.75	78.64	81.62	82.70	84.62	50.14	52.30	53.69	53.67	55.78	60.61	61.24	65.36	82.56	83.19	83.71	89.48	71.71	73.92	74.65	77.18
PDO		91.09	91.84	92.22	92.81	80.47	82.42	84.65	84.42	50.42	52.64	54.40	54.35	56.07	60.05	56.42	64.98	83.11	83.84	87.26	89.97	72.23	74.16	74.99	77.31
PDO-U		91.36	92.26	92.48	92.86	81.75	83.37	84.22	84.92	49.62	52.43	53.94	53.13	54.11	61.22	59.94	60.14	85.52	86.12	88.58	90.55	72.47	75.08	75.83	76.32
PDO-Up		91.43	92.26	92.46	92.96	81.75	83.37	84.48	84.92	51.60	54.35	54.46	54.49	55.63	62.14	61.98	68.25	85.56	86.12	88.01	90.63	73.19	75.65	76.28	78.25
DeMO		91.62	92.33	92.40	93.05	80.88	82.49	84.59	84.34	51.64	52.84	53.60	54.22	54.98	61.06	62.25	67.05	78.15	83.37	87.89	90.25	71.45	74.42	76.15	77.78
PLR-EMA		91.42	92.28	92.69	93.24	81.93	84.47	86.20	85.96	51.09	55.44	55.08	55.04	57.31	63.81	62.94	71.54	86.40	89.35	92.26	93.40	73.63	77.07	77.83	79.84
PLR-1		91.70	92.39	92.79	93.19	82.13	85.23	86.37	86.19	51.04	54.83	55.17	55.64	57.26	63.80	64.99	69.93	86.46	90.59	92.45	94.19	73.72	77.37	78.35	79.83
PLR-4		91.54	92.16	92.80	93.13	81.52	84.69	86.10	86.31	51.03	54.99	55.22	55.96	57.14	62.82	65.97	70.63	86.44	89.69	93.22	93.59	73.53	76.87	78.66	79.92

4 Experiments

We follow the standard evaluation protocol in prior work on ICL example ordering. For each dataset and each number of ICL examples $k \in \{4, 8, 16, 32\}$ we run 5 random seeds. In every seed, we (i) sample a fresh set of k demonstrations to include in the prompt, (ii) sample disjoint training and validation sets used only for training/selecting an ordering, and (iii) report performance on the (held-out) test set using the selected ordering. Additionally, we focus on the ordering of examples rather than their selection, ensuring that all methods use the same ICL examples so that the effects of ordering can be isolated. This strategy prevents unfair comparisons that could result from using different sets of examples. Our ablations are conducted on the SUBJ dataset, which is particularly sensitive to the choice and ordering of ICL examples.

4.1 Baselines

In this section we summarize the baselines we compare PLR against.

- `Static`. We keep the k sampled demonstrations in their original (data) order, i.e., no reordering.
- `Top-K`. We uniformly sample candidate permutations and evaluate them on the validation set. We use the same number of sampled candidates as PLR.
- `LocalE` / `GlobalE` (Lu et al., 2022b). Entropy-based probing for ordering selection. Given a candidate set of permutations, both methods rank permutations using the model's predictive entropy on a probing set and select the top-ranked ordering. `LocalE` uses the minimum entropy over the probing set (favoring permutations that yield at least one highly confident prediction), whereas `GlobalE` uses the average entropy (favoring permutations that are consistently confident).
- `PDO` (Xu et al., 2024). selects orderings using label-distribution criteria computed from the models predicted label probabilities. We report the three settings from the original work: `PDO-FewShot` (no additional data beyond the k demonstrations), `PDO-FewShotU` (additionally uses an unlabeled set), and `PDO-FewShotUP` (uses an unlabeled set and a prior label distribution).
- `DEmO` (Guo et al., 2024). performs a two-stage procedure: it first filters candidate orderings using a label-fairness criterion on content-free inputs, and then selects an ordering that maximizes an influence-based score for the test input (instance-level selection).
- PLR (ours: Section 3). We evaluate three variants of our approach:

Table 2: Test accuracy (%) averaged over seeds for `Qwen2.5-7B-instruct` on math benchmarks. best, second best, third best (per column). Avg is macro-average over GSM8K/DeepMath/Math500.

Method	GSM8K				DeepMath				Math500				Avg			
	k=4	k=8	k=16	k=32	k=4	k=8	k=16	k=32	k=8	k=4	k=32	k=16	k=4	k=8	k=16	k=32
`Static`	35.45	36.33	39.70	39.05	36.03	38.70	41.85	43.27	28.52	30.83	29.60	34.98	34.10	34.52	38.84	37.31
`Top-K`	36.05	38.86	39.35	40.40	36.49	38.74	42.20	45.13	31.40	31.13	29.80	38.60	34.56	36.33	40.05	38.44
`PLR-EMA`	40.88	39.50	39.85	41.60	36.64	41.52	42.52	46.16	32.32	33.73	31.80	39.00	37.08	37.78	40.46	39.85
`PLR-1`	39.63	39.65	39.85	42.85	36.64	39.93	42.69	45.62	31.43	32.70	32.00	39.00	36.32	37.00	40.51	40.16
`PLR-4`	40.88	39.77	41.40	42.80	36.64	39.17	42.46	46.36	31.90	33.73	33.40	38.78	37.08	36.95	40.88	40.85

- `PLR-EMA`: a single PL model updated via an exponential moving-average (EMA) rank update.
- `PLR-1`: a single PL model fit by maximum-likelihood estimation (MLE) on the elite set.
- `PLR-4`: a mixture of 4 PL components trained with an EM-style procedure.

We report the hyperparameters used for `PLR` in Appendix B.

4.2 Results

Classification. We evaluate on five classification benchmarks: MR (binary sentiment) (Pang and Lee, 2005), SST-5 (5-way sentiment) (Socher et al., 2013), TREC (question type) (Voorhees and Tice, 2000), AG's News (topic classification) (Zhang et al., 2015), and SUBJ (subjectivity) (Pang and Lee, 2004). Table 1 reports results for Qwen2.5-7B-Instruct (Yang et al., 2024a) and Llama-3.1-8B-Instruct (AI@Meta, 2024). Across datasets and number of ICL examples, `PLR` variants consistently rank among the top three methods on average. We note that the difference to `Top-K` is small when few (e.g. 4) ICL examples are present, since then the number of permutations is small. We get larger performance differences once the space of permutations cannot be sampled effectively anymore, e.g. for $k \geq 8$.

Reasoning. We additionally evaluate `PLR` on the mathematical reasoning benchmarks-GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), and DeepMath (He et al., 2025). Because these tasks have effectively unbounded answer spaces, label-probability based ordering methods such as `LocalE`/`GlobalE`, `PDO`, and `DEmO` are not directly applicable, so we compare against the Static and Top-K baselines. Table 2 reports results for `Qwen2.5-7B-Instruct`. Across all three benchmarks and number of ICL examples `PLR` consistently outperforms `Top-K`, indicating that learning a distribution over demonstration orders remains effective beyond classification and transfers naturally to open-ended generation and reasoning settings.

4.3 Ablations

Table 3: Results for SUBJ across different number of K of Mixture PL distribution and number of ICL examples k.

K	$k = 4$	$k = 8$	$k = 16$	$k = 32$
4	73.09	80.94	85.17	85.98
8	73.04	80.10	84.17	86.75
16	72.83	79.78	84.57	86.06
32	72.52	80.48	83.95	86.25
64	72.65	79.51	84.24	86.40

Table 4: SUBJ results for `PLR` variants with weighted vs. unweighted MLE.

Method	Weighting	k=4	k=8	k=16	k=32
PLR-1	Unweighted	72.80	80.33	83.44	85.90
	Weighted	72.78	80.88	83.85	87.48
PLR-4	Unweighted	73.09	80.94	85.17	85.98
	Weighted	73.09	80.70	84.22	86.97

Ablation: Number of PL components. We ablate the number of components K in the mixture-of-Plackett–Luce model to quantify how additional mixture capacity affects downstream performance. Intuitively, increasing K should allow the learned ordering distribution to better capture multi-modal structure (i.e., multiple distinct high-performing permutations). Table 3 reports results on SUBJ. Overall, performance is largely stable

Table 5: Results for SUBJ for varying parameters ρ of choosing the elite set in line 12 in Algorithm 1.

ρ	PLR-EMA				PLR-1				PLR-4			
	$k=4$	$k=8$	$k=16$	$k=32$	$k=4$	$k=8$	$k=16$	$k=32$	$k=4$	$k=8$	$k=16$	$k=32$
0.1	73.09	80.59	83.16	85.31	73.04	80.81	83.78	86.46	73.09	80.96	83.95	85.94
0.2	72.78	80.64	83.98	86.75	72.80	80.33	83.44	85.90	73.09	80.94	85.17	85.98
0.3	73.04	80.58	83.05	86.02	73.04	80.97	82.37	85.47	72.78	80.62	84.43	85.07
0.5	72.75	79.91	82.65	84.90	72.91	79.83	82.06	86.15	72.91	79.81	82.00	83.88

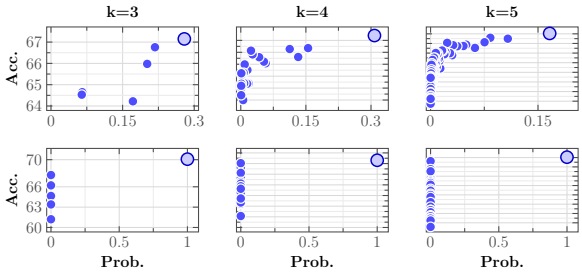

Figure 3: Top: Ablation probability-test accuracy results for PLR-EMA and Bottom: for PLR-1.

across K, with only modest fluctuations. This suggests that a small mixture (e.g., K=4) already captures the dominant modes of good orderings in this setting, and that further increasing K yields diminishing returns. We also see that potential overfitting occurs for larger K.

Ablation: Weighting in the MLE objective. We examine whether weighting elite permutations in the MLE refit improves performance. Concretely, we run PLR on SUBJ with both a single PL component and a K=4 mixture, comparing an *unweighted* MLE objective (all elite permutations contribute equally) against a *weighted* variant (elite permutations are reweighted by their training score). Results are shown in Table 4. Overall, the effect of weighting is inconsistent: depending on the number of demonstrations k (and the PL capacity), weighting can yield small gains or slight regressions. Given the lack of a clear, consistent advantage, we adopt the unweighted MLE objective in all main experiments for simplicity.

Ablation: Proportion of elites. We study how the elite fraction ρ (the proportion of sampled permutations retained for refitting in line 12 in Algorithm 1) affects performance. We run all three PLR variants with varying ρ, and report results in Table 5. If ρ is too large (e.g., 0.5), the elite set becomes too broad and the update signal is diluted. If ρ is too small (e.g., 0.1), the update becomes overly selective and less stable. In both cases, the quality of the training signal deteriorates. Based on this trade-off, we use $\rho = 0.2$ in all experiments as a robust default.

Ablation: Relation between probabilities and test accuracy. A benefit of modeling orders with a PL distribution is interpretability: the learned probability of a permutation can be compared directly to its downstream utility. We test whether higher-probability permutations also achieve higher test accuracy. For small $k \in \{3, 4, 5\}$, we can enumerate all $k!$ permutations, compute their PL probabilities, and evaluate their true test accuracy. We run this analysis for PLR-EMA and PLR-1, averaging over 5 random seeds. In Figure 3, permutations are sorted from most to least probable and we plot the corresponding probability and test accuracy trends. Across all k, we observe a clear monotonic relationship: permutations assigned higher probability tend to yield higher test accuracy. We also find that PLR-1 often concentrates mass on a single ordering (near-deterministic behavior), assigning probability close to 1 to the best permutation and near 0 to the rest, which is consistent with its MLE refitting objective and strong elite selection.

5 Conclusions

We have proposed an elegant probabilistic approach to choosing the ordering of ICL examples for prompting LLMs outperforming our reordering baselines. We believe that our probabilistic perspective can be extended to obtain principled approaches also to other related problems, for example simultaneously selecting and ordering ICL examples from a large set, selecting an instruction prompt and reordering the ICL examples, etc. Finally, reordering is largely absent from automatic prompt engineering approaches and we argue that this is another underexplored axis on how to improve automatic prompt engineering algorithms.

6 Limitations

While PLR achieves strong results across diverse benchmarks, we note several limitations.

Need for labeled data: PLR optimizes demonstration orderings using task-level metrics (e.g., accuracy), which requires labeled data to reliably score candidate permutations. This limits its direct applicability in fully unsupervised settings.

Task-specific optimization: In our current setup, PLR is optimized separately for each task. Developing a task-agnostic or transferable ordering strategy is an important direction for future work.

Scaling to larger models: Our experiments focus on 7B–8B models. It remains to be seen whether the improvements persist at larger scales and under different inference regimes.

Ethics Statement

We conducted this research in line with the ACL Code of Ethics and the ACM Code of Ethics and Professional Conduct. Because our method optimizes the ordering of in-context demonstrations, it can amplify positional and recency effects: examples placed earlier may disproportionately steer model outputs, potentially reinforcing spurious correlations, biased behaviors, or unsafe instructions present in the prompt. Order selection on a validation split can also introduce prompt-level overfitting and reduced robustness under distribution shift. To mitigate these risks, we use disjoint splits for ordering selection and final evaluation, report results averaged over multiple random seeds and shot counts, and recommend filtering demonstrations for sensitive or unsafe content before applying ordering optimization. More broadly, automated prompt optimization should be paired with robustness testing (including prompt-injection stress tests) and appropriate safeguards prior to deployment.

References

Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. 2024. Promptwizard: Task-aware prompt optimization framework. *arXiv preprint arXiv:2405.18369*.

AI@Meta. 2024. Llama 3 model card.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Pawel Batorski and Paul Swoboda. 2025a. Gps: General per-sample prompter. *arXiv preprint arXiv:2511.21714*.

Pawel Batorski and Paul Swoboda. 2025b. Piast: Rapid prompting with in-context augmentation for scarce training data. *arXiv preprint arXiv:2512.11013*.

Pawe Batorski, Adrian Kosmala, and Paul Swoboda. 2025. Prl: Prompts from reinforcement learning. *arXiv preprint arXiv:2505.14412*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Rahul Atul Bhope, Praveen Venkateswaran, KR Jayaram, Vatche Isahagian, Vinod Muthusamy, and Nalini Venkatasubramanian. 2025. Optiseq: Ordering examples on-the-fly for in-context learning. *arXiv preprint arXiv:2501.15030*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. *Preprint*, arXiv:2406.10248.

Anwoy Chatterjee, H S V N S Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. 2024. Posix: A prompt sensitivity index for large language models. *Preprint*, arXiv:2410.02185.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2023a. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. 2023b. On the relation between sensitivity and accuracy in in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 155–167.

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

Kwesi Adu Cobbina and Tianyi Zhou. 2025. Where to show demos in your prompt: A positional bias of in-context learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29548–29581.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Federico Errica, Giuseppe Siracusano, Davide Sanvito, and Roberto Bifulco. 2024. What did i do wrong? quantifying llms' sensitivity and consistency to prompt engineering. *arXiv preprint arXiv:2406.12334*.

Lizhe Fang, Yifei Wang, Khashayar Gatmiry, Lei Fang, and Yisen Wang. 2025. Rethinking invariance in in-context learning. In *The Thirteenth International Conference on Learning Representations*.

Yu Fei, Yifan Hou, Zeming Chen, and Antoine Bosselut. 2023. Mitigating label biases for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14014–14031.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.

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

Qi Guo, Leiyu Wang, Yidong Wang, Wei Ye, and Shikun Zhang. 2024. What makes a good order of examples in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14892–14904.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2310.08510*.

Xiaobo Guo and Soroush Vosoughi. 2025. Serial position effects of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 927–953.

Yinhan He, Wendy Zheng, Song Wang, Zaiyi Zheng, Yushun Dong, Yaochen Zhu, and Jundong Li. Hierarchical demonstration order optimization for many-shot in-context learning.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

Joonwon Jang, Sanghwan Jang, Wonbin Kweon, Minjin Jeon, and Hwanjo Yu. 2024. Rectifying demonstration shortcut in in-context learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4294–4321.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pages 3499–3508. PMLR.

Zekun Li, Baolin Peng, Pengcheng He, and Xifeng Yan. 2023. Evaluating the instruction-following robustness of large language models to prompt injection. *Preprint*, arXiv:2308.10819.

CHEN Liang, Li Shen, Yang Deng, Xiaoyan Zhao, Bin Liang, and Kam-Fai Wong. 2025. Pearl: Towards permutation-resilient llms. In *The Thirteenth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9).

Sheng Lu, Hendrik Schuff, and Iryna Gurevych. 2024a. How are prompts different in terms of sensitivity? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5833–5856.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022a. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *Preprint*, arXiv:2104.08786.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022b. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Yao Lu, Jiayi Wang, Raphael Tang, Sebastian Riedel, and Pontus Stenetorp. 2024b. Strings from the library of babel: Random sampling as a strong baseline for prompt optimisation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2221–2231.

R. Duncan Luce. 1959. *Individual Choice Behavior: A Theoretical Analysis*. John Wiley & Sons, New York.

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. *Advances in neural information processing systems*, 27.

Lucas Maystre and Matthias Grossglauser. 2015. Fast and accurate inference of plackett–luce models. *Advances in neural information processing systems*, 28.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Rethinking the role of demonstrations: What makes in-context learning work? *Preprint*, arXiv:2202.12837.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022b. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 11048–11064.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 271–278. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics.

Kha Pham, Hung Le, Man Ngo, and Truyen Tran. 2025. Rapid selection and ordering of in-context demonstrations via prompt embedding clustering. In *The Thirteenth International Conference on Learning Representations*.

R. L. Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2):193–202.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. 2025. Benchmarking prompt sensitivity in large language models. In *European Conference on Information Retrieval*, pages 303–313. Springer.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 CHI conference on human factors in computing systems*, pages 1–7.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2655–2671.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *Preprint*, arXiv:2310.11324.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, and 1 others. 2022. On the effect of pretraining corpora on in-context learning by a large-scale language model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5168–5186.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023a. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. *Advances in Neural Information Processing Systems*, 36:15614–15638.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. 2023b. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. 2024. Prompt optimization with ease? efficient ordering-aware automated selection of exemplars. *Advances in Neural Information Processing Systems*, 37:122706–122740.

Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2024. Addressing order sensitivity of in-context demonstration examples in causal language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6467–6481.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Zhichao Xu, Daniel Cohen, Bei Wang, and Vivek Srikumar. 2024. In-context example ordering guided by label distributions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2623–2640.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024b. Large language models as optimizers. *Preprint*, arXiv:2309.03409. ICLR 2024.

Zhao Yang, Yuanzhe Zhang, Dianbo Sui, Cao Liu, Jun Zhao, and Kang Liu. 2023. Representative demonstration selection for in-context learning with two-stage determinantal point process. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5443–5456.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Kaiyi Zhang, Ang Lv, Yuhan Chen, Hansen Ha, Tao Xu, and Rui Yan. 2024. Batch-icl: Effective, efficient, and order-agnostic in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10728–10739.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021a. Calibrate before use: Improving few-shot performance of language models. *Preprint*, arXiv:2102.09690.

Zhibing Zhao, Peter Piech, and Lirong Xia. 2016. Learning mixtures of Plackett-Luce models. In *International Conference on Machine Learning*, pages 2906–2914. PMLR.

Zhibing Zhao and Lirong Xia. 2019. Learning mixtures of Plackett-Luce models from structured partial orders. *Advances in Neural Information Processing Systems*, 32.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. Pmlr.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed H.

Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. 2024. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. *Preprint*, arXiv:2306.04528.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. Prosa: Assessing and understanding the prompt sensitivity of llms. *Preprint*, arXiv:2410.12405.

A Expressivity of Mixture-PL

Theorem 1 (Expressivity: Mixture-PL is dense, single PL is not (for $n \geq 3$))**.** *Let $\Delta(S_n)$ denote the set of all probability distributions over permutations S_n equipped with the ℓ_1 norm $\|p - q\|_1 = \sum_{\pi \in S_n} |p(\pi) - q(\pi)|$.* ***(i) Density of mixtures.*** *For any target distribution $p \in \Delta(S_n)$ and any $\varepsilon > 0$, there exist an integer $K \leq n!$, weights $\alpha_1, \ldots, \alpha_K$ with $\alpha_k \geq 0$ and $\sum_{k=1}^{K} \alpha_k = 1$, and logits $\theta^{(1)}, \ldots, \theta^{(K)} \in \mathbb{R}^n$ such that the mixture*

$$q(\pi) \;=\; \sum_{k=1}^{K} \alpha_k \, \mathrm{PL}(\pi \mid \theta^{(k)})$$

satisfies $\|q - p\|_1 < \varepsilon$. In particular, the family of mixture-PL models is dense in $\Delta(S_n)$.

(ii) Non-density of a single PL component (when $n \geq 3$). *For $n \geq 3$, the single-component PL family*

$$\mathcal{F}_{\mathrm{PL}} = \{\mathrm{PL}(\cdot \mid \theta) : \theta \in R^n\}$$

is not dense in $\Delta(S_n)$.

B Hyperparameters

In this section we give the exact hyperparameters we run our method with.

C Usage of LLMs

We used LLMs to improve the clarity and style of the writing and presentation, and to assist with drafting parts of the implementation code. All substantive research decisions and the core technical contributions were made by the authors.

Table 6: Hyperparameters used for PLR (unless stated otherwise). We use the same notation as in the main body/Alg. 1: T (CE iterations), B (samples/iter), ρ (elite fraction), K (final draws), α (EMA smoothing), and τ (rank temperature). For the mixture model, K also denotes the number of mixture components (as in the mixture definition in Sec. 3).

Hyperparameter	Symbol	Value	Meaning
Experimental protocol			
# demonstrations	k	$\{4, 8, 16, 32\}$	Shots used in the prompt.
# seeds	—	5	Runs per dataset/k/method.
Validation budget	$\lvert D_{\text{val}}\rvert$	1000	Total labeled pool used for ordering selection (if using a subset; otherwise `full`).
Inner/outer split	—	$80/20$	Inner set used for updates (D_t^{train}); outer set used for final selection (D^{val}).
Scoring batch size	—	16	Batch size for label-scoring / evaluation.
`PLR (shared CE / distributional optimization)`			
CE iterations	T	15	Number of distribution-update iterations.
Samples per iteration	B	15	Permutations sampled per iteration from the current distribution.
Elite fraction	ρ	0.2	Top-$\lceil \rho B \rceil$ permutations retained as elites.
Final draws	K	10	# permutations sampled from the *final* distribution for MC selection on D^{val}.
EMA smoothing	α	0.7	Smoothing used in the update (heuristic rank update and as optional smoothing for MLE/mixture refits).
Rank temperature	τ	1.0	Rank-to-logit temperature in the heuristic update $\theta_i^{\text{tgt}} = -\bar{r}_i/\tau$.
`PLR-1 (single PL, MLE refit on elites)`			
Adam steps	—	60	Gradient steps for PL MLE on the elite permutations.
Learning rate	—	0.1	Adam learning rate for PL MLE.
L2 penalty	—	0.0	L2 regularization coefficient in the MLE objective.
Logit clip	—	20	Clamp logits to $[-c, c]$ during MLE fitting for stability.
Weighted elites	—	OFF	If ON, weight elite permutations by training score in the MLE fit.
`PLR-4 (mixture of PL, EM-style refit on elites)`			
# mixture components	K	4	Number of PL components in the mixture.
Min component weight	—	10^{-3}	Lower bound on mixture weights during training (stability).