Found in the Middle: Permutation Self-Consistency Improves Listwise Ranking in Large Language Models

Raphael Tang,[*1] **Xinyu Zhang,**[*2] **Xueguang Ma,**[2] **Jimmy Lin,**[2] **Ferhan Ture**[1]
[1]Comcast Applied AI [2]University of Waterloo
[1]{raphael_tang, ferhan_ture}@comcast.com [2]{x978zhang, x93ma, jimmylin}@uwaterloo.ca

Abstract

Large language models (LLMs) exhibit positional bias in how they use context, which especially complicates listwise ranking. To address this, we propose *permutation self-consistency*, a form of self-consistency over ranking list outputs of black-box LLMs. Our key idea is to marginalize out different list orders in the prompt to produce an order-independent ranking with less positional bias. First, given some input prompt, we repeatedly shuffle the list in the prompt and pass it through the LLM while holding the instructions the same. Next, we aggregate the resulting sample of rankings by computing the central ranking closest in distance to all of them, marginalizing out prompt order biases in the process. Theoretically, we prove the robustness of our method, showing convergence to the true ranking in the presence of random perturbations. Empirically, on five list-ranking datasets in sorting and passage reranking, our approach improves scores from conventional inference by up to 7–18% for GPT-3.5 and 8–16% for LLaMA v2 (70B), surpassing the previous state of the art in passage reranking. Our code is at https://github.com/castorini/perm-sc.

1 Introduction

Large language models (LLMs) respond cogently to free-form textual prompts and represent the state of the art across many tasks (Zhao et al., 2023). Their quality, however, varies with nuisance positional factors such as prompt order and input length. As a descriptive example, consider this prompt:

> *Arrange the following passages in decreasing relevance to the query, "what are shrews?"*
>
> *(1) Cats hunt small mammals, such as shrews ...*
> *(2) Shrews are mole-like mammals, widely ...*
> *(3) Shrews use their noses to find prey and ...*

The correct output order is $(2, 3, 1)$, from most relevant to least, but several positional biases may interfere with the model. Liu et al. (2023) demonstrate that LLMs tend to get "lost in the middle" of a long context and use the middle portion poorly, which suggests that the middle passage (2) in the example may get misranked (e.g., $3, 1, 2$). Wang et al. (2023a) find prompt order to affect quality, with some orders outperforming others; if items 1 and 3 were swapped in the prompt, the LLM would perhaps generate the mistaken ranking $(2, 1, 3)$.

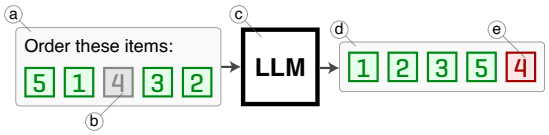


Figure 1: The conventional decoding process for listwise ranking with input prompt ⓐ, language model ⓒ, and output ranking ⓓ. The grey item ⓑ is "lost in the middle" by the LLM, resulting in its misranking ⓔ.

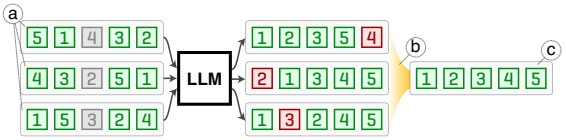


Figure 2: Our permutation self-consistency process. With the instruction fixed, we shuffle the input list for prompts ⓐ, producing outputs with different mistakes. We then aggregate ⓑ these output rankings into one ⓒ.

In this paper, we mitigate positional biases for listwise-ranking LLMs. We propose *permutation self-consistency*, a novel decoding strategy for improving the quality, consistency, and prompt-order invariance of black-box LLMs. First, we construct prompts with randomly permuted input lists, from which the LLM generates a set of output rankings. Then, we aggregate these outputs into the central ranking that minimizes the Kendall tau distance to all of them, marginalizing out prompt order as a factor; see Figures 1 and 2. As related work, Stoehr et al. (2023) train order-aware probes on the latent representations of language models to increase consistency, but they assume white-box model access, whereas we do not.

[*]Equal contribution.

Next, we assess the effectiveness of permutation self-consistency, both theoretically and empirically. Theoretically, we prove that it recovers the true ranking under arbitrary noise distributions, with enough observations and at least one correctly ordered pair in each observation. Experimentally, we apply our method to tasks in math and word sorting, sentence ordering, and passage reranking, consistently increasing the scores of GPT-3.5, GPT-4, and LLaMA v2 (70B; Touvron et al., 2023) by up to 4–17%, 9–24%, and 8–16%, respectively. On TREC-DL19 and TREC-DL20 (Craswell et al., 2020, 2021), two passage ranking datasets, we establish the new state of the art. From this evidence on multiple tasks, we conclude that permutation self-consistency improves listwise ranking in LLMs, which is partially influenced by positional bias, as shown in Section 3.2.

Finally, we conduct auxiliary analyses to justify our design choices. In Section 4.1, our hyperparameter study finds that quality quickly rises with the number of aggregated output rankings: the score improvement from using five aggregated rankings reaches 67% of twenty, on average, suggesting that a few suffice for quality gain. We further demonstrate that sampling temperature is ineffective for us, unlike the original self-consistency work (Wang et al., 2023b) in chain-of-thought reasoning, likely because listwise ranking does not require exploration of various reasoning paths.

Our contributions are as follows: **(1)** we propose a novel decoding technique for improving the quality, consistency, and position invariance of black-box, listwise-ranking LLMs; **(2)** we empirically establish the new state of the art in passage reranking and theoretically prove the robustness of our method to certain classes of ranking noise, including "lost-in-the-middle" type ones; and **(3)** we provide new analyses on positional biases in listwise-ranking LLMs, finding that these biases depend on pairwise positions of items in the list.

2 Our Approach

2.1 Preliminaries

Notation. We define an *n-ranking* as a permutation $\sigma : \{1, \ldots, n\} \mapsto \{1, \ldots, n\}$. For some sequence $\boldsymbol{X} := \{X_i\}_{i=1}^n$, define $\boldsymbol{X}[\sigma]$ as the permuted sequence of $\boldsymbol{X}$ transformed by σ, where $\boldsymbol{X}[\sigma]_i := X_{\sigma(i)}$. Let the inversion vector of σ be

$$\mathrm{inv}(\sigma)_i := \#\{j : \sigma(j) > \sigma(i), j < i\}. \quad (1)$$

To quantify dissimilarity, the Kendall tau distance between two rankings σ_1 and σ_2 is the number of inversions in $\sigma_1^{-1} \circ \sigma_2$:

$$d_\kappa(\sigma_1, \sigma_2) := \sum_{i=1}^{n} \mathrm{inv}(\sigma_1^{-1} \circ \sigma_2)_i. \quad (2)$$

In other words, it is the number of pairwise disagreements, or *discordant* pairs, in the permutation ordering. The distance is one affine transform away from the Kendall tau correlation, used to measure list order similarity (Kendall, 1948):

$$\tau(\sigma_1, \sigma_2) := 1 - \frac{2d_\kappa(\sigma_1, \sigma_2)}{\binom{n}{2}}. \quad (3)$$

In the extreme, $\tau = 1 \iff \sigma_1 = \sigma_2$, and $\tau = -1$ implies that one is the other's reverse.

2.2 Permutation Self-Consistency

How do we mitigate positional biases in listwise-ranking LLMs? We find inspiration in the self-consistency framework (Wang et al., 2023b), which improves quality and consistency in chain-of-thought prompting (Wei et al., 2022). The approach has two main stages: first, it *samples* multiple answers for an input prompt; then, it *aggregates* the sampled answers into a single, high-quality one, hence "marginalizing out" separate reasoning paths from the language model.

Unfortunately, self-consistency does not readily generalize to listwise ranking for a few reasons. For one, it is limited to point predictions, greatly simplifying the aggregation procedure to taking the majority vote. For another, sampling temperature, the method's mainstay of generating diverse samples for aggregation, has little effect on (and at times harming) the quality of aggregated predictions in listwise ranking, as shown in Section 4.1. Lastly, self-consistency does not explicitly address positional bias, the central issue of our paper.

Nevertheless, its shuffle–aggregate paradigm is still a useful template. With it, we propose *permutation self-consistency*: for the first sample step, we randomly shuffle the list in the prompt to curate a diverse set of rankings, each with different position biases. For the next aggregate step, we compute the central ranking closest in Kendall tau distance to all the sampled rankings, which, like self-consistency, marginalizes out the independent variable (in the original, reasoning paths; in ours, prompt order). Intuitively, we intervene on list order, collect output rankings, then aggregate, breaking the association between individual list order and output rankings.

Task	Example Input Prompt
Math Sorting	Sort these expressions: 3 / 2, 1 - 5, ...
Sentence Ordering	Order the shuffled sentences: [1] The...
Passage Ranking	Order these by relevance to the query, "what are shrews?": [1] Cats hunt...

Table 1: Listwise-ranking input prompt examples.

Formally, we are given an input sequence of items $\boldsymbol{X} := \{X_i\}_{i=1}^n$, such as a list of passages, along with a listwise-ranking LLM $h(\boldsymbol{X}; s)$ that returns an n-ranking on some string prompt s; see Table 1 for an example. First, we construct a diverse set of output rankings by randomly permuting $\boldsymbol{X}$ and passing it through the LLM, like how self-consistency uses temperature to vary their output. Specifically, we sample a sequence

$$\hat{\sigma}_i := h(\boldsymbol{X}[\pi_i]; s) \text{ for } 1 \leq i \leq m, \tag{4}$$

where π_i is drawn uniformly at random from the set of all possible n-rankings. As noted previously, each output ranking has positional bias, but mistakes are expected to differ among the outputs because of our input order randomization. We then "marginalize out" these individual biases by aggregating the output rankings into a single central ranking. One method with attractive theoretical properties is the Kemeny–Young (Kemeny, 1959) optimal ranking of the outputs—that is, the central ranking that minimizes the sum of its Kendall tau distances to every output ranking:

$$\bar{\sigma} := \operatorname*{argmin}_{\sigma} \sum_{1 \leq i \leq m} d_\kappa(\hat{\sigma}_i, \sigma). \tag{5}$$

Our approach returns $\bar{\sigma}$ as the prediction for $\boldsymbol{X}$ and terminates. Although this calculation is NP-hard, fast exact and approximate algorithms exist (Conitzer et al., 2006; Ali and Meilă, 2012), many implemented in our codebase.

Passage reranking. The task of passage ranking ranks a set of provided passages in order of relevance to a given query. The use of permutation self-consistency for this case deserves special attention. Due to the LLM input length constraint, predominant LLM-based approaches such as RankGPT (Sun et al., 2023), LRL (Ma et al., 2023), and RankVicuna (Pradeep et al., 2023) stride the LLM across fixed windows of items from the back of the list to the front, rather than output a ranking in a single pass. In this case, we simply apply permutation self-consistency to each window.

2.3 Theoretical Guarantees

We now show that for certain kinds of noisy rankings, the Kemeny ranking can recover the true ranking given enough observations. For example, if there always exists some random pair of items that are correctly ranked among randomly ordered observations, we will converge to the true ranking.

Definition 2.1. *For two rankings σ_1 and σ_2, the concordant subset is a set S' where $\forall i$ and $j \in S', \sigma_1(i) < \sigma_1(j) \wedge \sigma_2(i) < \sigma_2(j)$ or $\sigma_1(i) > \sigma_1(j) \wedge \sigma_2(i) > \sigma_2(j)$.*

Proposition 2.1. *Let there be a true ranking σ and a sequence of noisy rankings $\hat{\boldsymbol{\sigma}} := \{\hat{\sigma}_i\}_{i=1}^m$. Suppose each noisy ranking has a uniformly random, nonempty concordant subset S' with σ, and the remaining rank elements not in S' represent a random permutation. Then the Kemeny–Young ranking $\bar{\sigma}$ of $\hat{\boldsymbol{\sigma}}$ converges in probability to σ, i.e., it is a consistent estimator.*

Proof sketch. Let A_{ij} be the event that the sum of discordant pairs indexed by i and j across each ranking in $\hat{\boldsymbol{\sigma}}$ is greater than the number of concordant ones. $\mathbb{P}(A_{ij})$ is upper-bounded by $O(\frac{1}{m})$. The union bound of $\mathbb{P}(\bigcap_{i,j} A_{ij})$ shows that the probability of the sum of discordant pairs being greater than that of the concordant pairs vanishes for any pair as m approaches infinity. Thus, the Kemeny-optimal ranking will always approach σ for $m \to \infty$, concluding our proof. □

To extend this result, we demonstrate that, in the presence of any arbitrary distribution of ranking noise (e.g., the hypothetical "lost-in-the-middle" kind), characterized empirically in Section 3.2, our approach yields a consistent estimator for the true ranking, given that at least one possibly nonrandom pair of items is always concordant:

Proposition 2.2. *Let there be a true ranking σ, input ranking σ_{in}, and a ranking noise distribution $\mathbb{P}(\sigma_{noisy}|\sigma_{in})$, where σ_{noisy} always has a (possibly nonuniform) nonempty concordant subset S' with σ. Then the permutation self-consistency procedure is a consistent estimator of σ when applied to σ_{in} as the input and LLM parameterized by $\mathbb{P}(\sigma_{noisy}|\sigma_{in})$.*

Proof sketch. Observe that the first shuffling stage of permutation self-consistency transforms the premises into those of Proposition 2.3. Since the next stage of the method involves the same Kemeny–Young ranking as the proposition does, the rest of the proof quickly follows. □

1. MathSort: Sort ten arithmetic expressions by value.
Example: Sort the following expressions from smallest to largest: `3 / 5, 2 - 9, 6 * 5, 2 * 1, 3 / 1, 9 * 9, 1 - 9, 9 + 8, 3 / 5, 1 / 9`. The output format should be a comma-separated list containing the exact expressions; do not reduce them. Only respond with the results; do not say any word or explain.
2. WordSort: Order ten words alphabetically.
Example: Order these words alphabetically: `aaron`, `roam`, `aardvark`, `nexus`, [...]. The output format should [...]
3. GSM8KSort: Unscramble sentences from GSM8K.
Example: Order the scrambled sentences logically: - `She took 1 hour to walk the first 4 miles` [...] - `Marissa is hiking a 12-mile trail.` - `If she wants her average speed to be 4` [...] The output format should have each sentence on a new line. Only respond with the results; do not say any [...]

Table 2: Example prompts for our three sorting tasks.

Method	MATHSORT		WORDSORT		GSM8KSORT	
	Orig.	PSC	Orig.	PSC	Orig.	PSC
$LLaMA_2$-7B	8.7	24.2	41.3	59.9	6.1	21.3
$LLaMA_2$-13B	16.7	26.0	65.4	78.8	42.7	46.8
$LLaMA_2$-70B	27.9	31.3	74.6	81.0	61.1	71.2
GPT-3.5	64.0	75.2	85.9	88.1	82.1	88.4
GPT-4	83.5	**89.6**	89.9	**92.0**	88.4	**90.5**

Table 3: Kendall tau correlation scores on our sorting tasks. Original scores are the median across 20 single runs, and PSC aggregates those 20. Underline indicates improvement from PSC and bold denotes best.

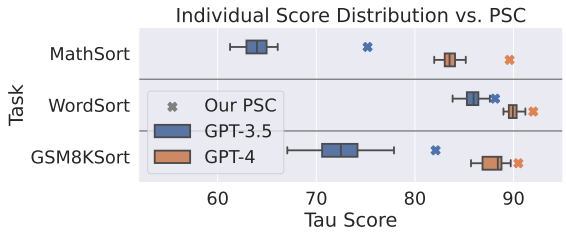


Figure 3: The distribution of sorting task scores from twenty individual runs plotted against our PSC score. Our PSC outperforms the best of any individual run.

3 Experiments

We conduct experiments on sorting and passage ranking, which constitute two distinct types of problems in listwise ranking.

3.1 Sorting Tasks

Setup. We build three functionally distinct datasets called MathSort, WordSort, and GSM8KSort, corresponding to numerical sorting, alphabetical ordering, and sentence arrangement, respectively. For MathSort, the task is to sort ten random mathematical expressions of the form `digit op digit`, where `digit` is a single digit and `op` is one of `+`, `-`, `*`, or `/`. In WordSort, the goal is to order ten random English words alphabetically. Finally, GSM8KSort is a sentence-unscrambling task over the test set of the GSM8K reasoning dataset (Cobbe et al., 2021). For consistency and tractability, we use 100 examples in each dataset; see Table 2 for prompts.

Although less practical than passage ranking, these synthetic sorting datasets have certain advantages. The items are intrinsically comparable, especially in MathSort and WordSort, whose elements have unequivocal order (e.g., "aardvark" must precede "abacus" in WordSort). On the other hand, passage ranking relies on human judgment, where label noise may confound findings. Synthetic construction also enables control of item length: MathSort examples are fixed at three tokens, WordSort at a single word, and GSM8K one sentence.

For our LLMs, we choose the open family of LLaMA v2 models (Touvron et al., 2023) and the closed GPT-3.5 (Turbo, the "0613" version) and GPT-4 from OpenAI, both the state of the art. We apply permutation self-consistency with $m = 20$ output rankings, resulting in 20 parallel calls to the LLM per example.

Results. We present our main results in Table 3, naming our method "PSC" for short. PSC consistently outperforms conventional inference on all three datasets and five models by an average of 42% in Kendall tau correlation, with gains skewed toward the smaller $LLaMA_2$ variants. Specifically, $LLaMA_2$-7B, 13B, and 70B attain average score increases of 157%, 28%, and 12%, respectively, while GPT-3.5 and GPT-4 improve by 3–18% and 2–7%. We attribute this to the already high quality of the larger 70B and GPT models, which leave less room for improvement. We conclude that PSC improves listwise ranking on sorting tasks, with higher gains on lower-quality models.

One foreseeable question is whether any individual runs surpass PSC, which would weaken the case for rank aggregation. To answer this, we plot the distribution of the individual scores against PSC in Figure 3. We observe that PSC reliably beats all individual runs by 1–12%, improving the most on tasks and models with lower baseline quality, such as MathSort and GPT-3.5. These findings bolster the necessity of the aggregation step.

First Stage	Top-k	Method	TREC-DL19		TREC-DL20	
			Original	Our PSC	Original	Our PSC
None	All	(1) BM25	50.58	–	47.96	–
	All	(2) SPLADE++ ED	73.08	–	71.97	–
Supervised Approaches						
BM25	100	(3) MonoT5 (T5-3B)	71.83	–	68.89	–
	100	(4) RankT5 (T5-3B)	71.22	–	69.49	–
Unsupervised Approaches						
BM25	100	(5) PRP-Best (FLAN-T5-XXL)	69.87	–	69.85	–
	100	(6) PRP-Best (FLAN-UL2)	72.65	–	70.68	–
	100	(7) RankVicuna	66.83	68.70	65.49	65.68
	20	(8) Single (GPT-3.5)	60.95 (60.96)	61.49	57.64 (57.68)	59.62
	20	(9) Single (GPT-4)	60.88 (60.92)	64.88	57.78 (57.89)	62.49
	100	(10) RankGPT (GPT-3.5)	68.00 (68.13)	70.77	62.08 (63.20)	62.70
	100	(11) RankGPT (GPT-4)	75.00 (75.59)	**75.66**	70.36 (70.56)	**71.00**
SPLADE++ ED	100	(12) RankVicuna	74.59	74.13	74.73	74.06
	20	(13) Single (GPT-4)	73.21 (73.36)	**76.87**	71.97 (73.63)	**78.52**
	100	(14) RankGPT (GPT-4)	74.64 (74.93)	76.01	70.76 (71.08)	75.14

Table 4: nDCG@10 results on TREC-DL19 and TREC-DL20. Scores in parentheses are the maximum across three runs, while those outside the median. Improvements from PSC are underlined and best per-section scores are bolded. According to the one-tailed signed-rank test, paired differences between the original and PSC are statistically significant at the 99% confidence level ($p < 0.01$).

3.2 Passage Reranking Task

For a more applied case, we evaluate our method on passage reranking. In this task, we are given a query and an initial list of relevant documents from a fast, first-stage retriever. We must then reorder these documents to improve their final relevance.

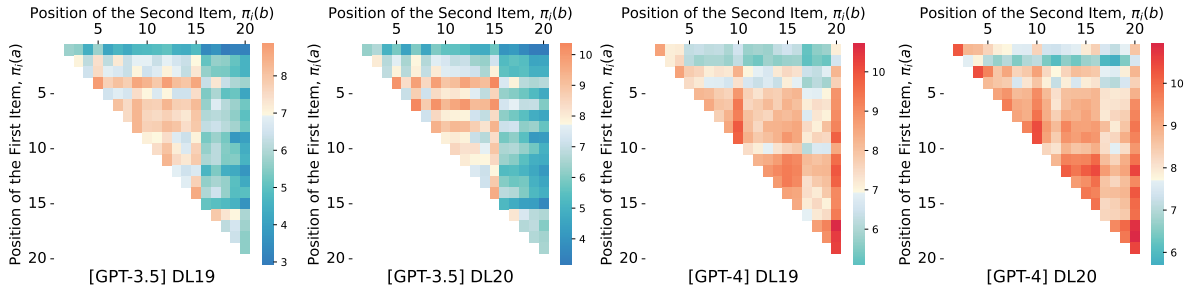
Setup. From the TREC Deep Learning Track, we select the two passage retrieval test sets from TREC-DL19 and TREC-DL20 (Craswell et al., 2020, 2021), both canon in the literature (Pradeep et al., 2023; Qin et al., 2023). These datasets are built on the MS MARCO v1 corpus (Bajaj et al., 2016), which contains 8.8 million passages. As is standard, we rerank the top-100 passages retrieved by the first-stage BM25 (Robertson et al., 2009) or SPLADE++ EnsembleDistill (ED; Formal et al., 2021), reporting nDCG@10 scores for quality.

Like the sorting tasks, we pick one open LLM, RankVicuna (Pradeep et al., 2023), fine-tuned from Vicuna-7B (Chiang et al., 2023), and one closed family, GPT-3.5 and GPT-4—all models are the present state of the art. RankVicuna and GPT-3.5 have matching context lengths of 4096, half of GPT-4's 8192. We similarly apply permutation self-consistency with $m = 20$ runs. Furthermore, for three of our variants named "single," we reduce the top-100 to 20 and discard the windowing strategy used in RankGPT and RankVicuna, described in Section 2.2. This allows us to fit all passages in a single call and thus remove potentially confounding interactions between the windowing method and permutation self-consistency.

For our supervised baselines, we report results from the MonoT5 (Nogueira et al., 2020) and RankT5 (Zhuang et al., 2023) models, based on the T5 language model (Raffel et al., 2020). For the unsupervised baselines, we copy figures from the state-of-the-art pairwise ranking results across the variants in Qin et al. (2023), which we name PRP-Best for short.

Results. We present our results in Table 4. With PSC, we establish four state-of-the-art results: first, a new best in BM25 for DL19 (row 11), edging ahead of the prior record from RankGPT by 0.07 points; second, the same for DL20 (row 11), leading PRP by 0.32 points (row 6); third, the overall top result on DL19 of 76.87 from SPLADE++ (row 13), outperforming the previous by 1.28 (row 11); and fourth, the state of the art of 78.52 on DL20 (row 13), a 3.79-point increase over the previous best from RankVicuna (row 12).

Overall, our PSC approach consistently improves ordinary decoding and beats the maximum individual score across three runs (see scores in parentheses), yielding gains on 13 out of 16 model–dataset combinations (see PSC columns in rows 7–14). On average, RankVicuna, GPT-3.5, and GPT-4 see relative score increases of 0.4%, 2%, and 5% with PSC. Mixed results on RankVicuna



(a) Single (GPT-3.5) on DL19 and DL20.

(b) Single (GPT-4) on DL19 and DL20.

Figure 4: Distribution of "reversions" after reranking. Blues are below the observed dataset average and reds above the average. For two input list positions $i \in [1, 20]$ and $j \in (i, 20]$, i indexes the rows and j the columns. For example, the cell at $(1, 2)$ is the reversion of the first two input items across the dataset. Note that highly saturated colors indicate over- and under-reversion *relative* to other pairs in the dataset rather than in the absolute sense.

likely result from its inherent robustness to positional bias, instilled by its training process that uses random shuffling as part of data augmentation; thus, the shuffling step from PSC has less effect.

The choice of the first-stage reranker has a clear impact, with SPLADE++ adding an average of 7.26 points over the corresponding BM25 models. In fact, reranking the top-20 SPLADE items (row 13) in a single call outperforms doing the top-100 (row 14) using a sliding call window. We conjecture that this results from imperfections in the RankGPT windowing algorithm, which shows especially for strong retrievers, where the top-20 already contains many relevant documents.

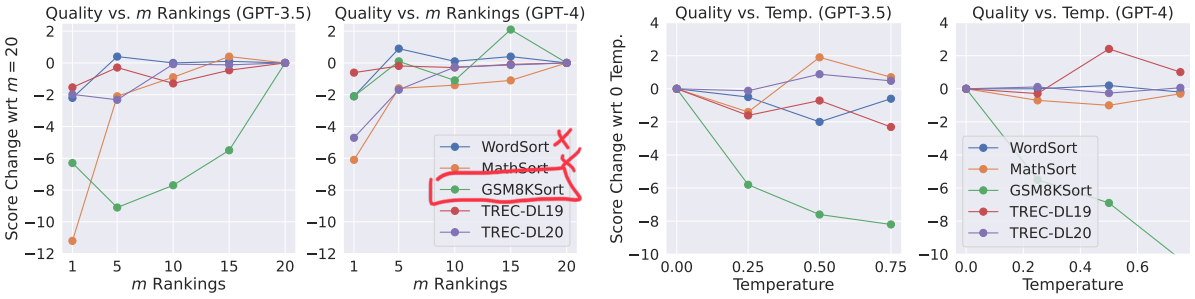
Finally, we note one particularly intriguing phenomenon: in the top-20 single-call setting, GPT-3.5 and GPT-4 have similar baseline quality without PSC (rows 8 and 9, first column in each group), but PSC boosts GPT-4 more than GPT-3.5 (row 9, second columns). As we explore in depth next, this possibly results from GPT-4 being more "equally biased" across the item positions and hence providing PSC more useful rankings for aggregation.

Positional bias analysis. We analyze how list order bias varies with the input positions on the "single" GPT models for BM25 (from Table 3, rows 8 and 9), which avoid confounds from RankGPT's window strategy. The design of our analysis is as follows, with notation mirroring Section 2.2: consider the item pair (X_a, X_b) with input list positions $(\pi_i(a), \pi_i(b))$, where $\pi_i(a) < \pi_i(b)$ for some random permutation π_i. If the output positions satisfy $\hat{\sigma}_i(a) > \hat{\sigma}_i(b)$ after reranking, we say the order is reversed, and we call the sum of reversed pairs per data point *"reversions."* In Figure 4, we visualize the distribution of reversions by input position pair, with $\pi_i(a)$ as the y-axis and $\pi_i(b)$ as the x-axis, whose positions range from 1–20 for each of the top-20 passages. For cross-model comparability, we normalize by dataset.

Under the null hypothesis of no positional bias, the distribution of reversions should be uniform because the input lists are randomly permuted, which severs any association between input order and output ranking. However, Figure 4 contradicts this. Prominently, the center of Figure 4a is redder than the edges, indicating that pairs with both items closer to the middle are reversed more often by GPT-3.5 than those at the start and the end of input lists. In Figure 4b, bottom areas are also more red than the top, showing that pairs with items at the end of the list are more frequently reversed by GPT-4 than pairs at the start are.

Other subtle patterns emerge upon examination. First, in Figure 4a, a dark block appears after column 15, suggesting that GPT-3.5 does *not* focus well on items past the fifteenth. Second, the colors interleave in a grid pattern across both columns and rows—possibly an artifact of its pretraining. We conclude that different positional biases exist in reranking LLMs, varying by model and dataset.

The analysis also helps to explain our prior experimental results. Comparing Figure 4a and 4b, we observe that GPT-4 generally reverses more pairs than GPT-3.5 and is closer to the optimal number of reversals, thus providing higher quality to the aggregated rankings. This may explain why PSC benefits GPT-4 (single) more than it does GPT-3.5 (single), i.e. row 9 vs. row 8 in Table 4. Similarly, both models tend to reverse more pairs on DL20 than on DL19, and results also indicate that PSC improves DL20 more than it does DL19.



(a) Quality vs. number of output rankings ($\rho = 0.17$).

(b) Quality vs. text generation temperature ($\rho = -0.078$).

Figure 5: Quality across all datasets for various choices of aggregate size and temperature. For output rankings, we use $m = 20$ as our frame of reference; for temperature, 0.0. In the subfigure captions, ρ denotes Spearman's rho.

4 Sensitivity Analyses

In this section, we investigate the importance of each component of permutation self-consistency to justify our modeling choices.

4.1 Hyperparameter Studies

Output rankings. Throughout the paper, we espoused aggregating over $m = 20$ output rankings, but is more actually better? If, say, five outperforms twenty, we could decrease the number of parallel calls to the model, conceivably saving cost. To answer this question, we sweep the aggregate size between one and twenty across all datasets, plotting the resulting score differences from using the default twenty. We pick GPT-3.5 and GPT-4 as our target models, as they are used in all tasks.

We plot our results in Figure 5a. On both models, we find that output quality rapidly converges to that of using the full twenty, five being 67% as effective on average. The score averages increase monotonically with the number of rankings ($\rho = 0.17$), with GSM8KSort on GPT-3.5 as an outlier (left subplot), possibly because of output variance—the next study on sampling temperature shows that it is highly sensitive to randomness. We conclude that picking $m = 20$ output rankings is effective, though returns sharply diminish after 5–10.

Sampling temperature. Self-consistency (Wang et al., 2023b) uses temperature as their sampling strategy to produce different outputs to aggregate over, but it is ineffective for us, perhaps because listwise ranking does not admit multiple reasoning paths like chain-of-thought prompting does. To assess this rigorously, we vary the temperature between 0 and 0.75, following the original method's 0.5–0.7 (Wang et al., 2023b). For consistency, we use the same setup from before and fix $m = 20$.

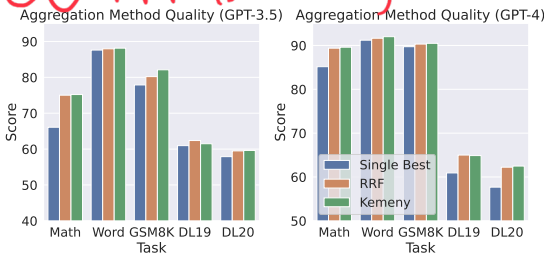


Figure 6: Scores for the alternative reciprocal rank fusion (RRF) and our Kemeny rank aggregation method.

We plot our results in Figure 5b. Temperature has little effect on the quality ($\rho = -0.078$), again with GSM8KSort as an outlier, where the extra randomness drastically hurts quality on both models. This sensitivity to randomness is also evident in Figure 3, where GSM8K has the widest interquartile range of the tasks. In conclusion, this evidence grounds our choice of not using temperature.

4.2 Rank Aggregation Comparison

Reciprocal rank fusion (RRF; Cormack et al., 2009) is a state-of-the-art alternative to our chosen Kemeny ranking method. It sorts items by the score

$$\text{RRFScore}(X_j) := \sum_{1 \le i \le m} \frac{1}{k + \hat{\sigma}_i(j)} \qquad (6)$$

for each item X_j, rankings $\hat{\sigma}_i$, and $k = 60$. RRF had been under our consideration, but we picked Kemeny ranking for its theoretical robustness and empirical effectiveness. Shown in Figure 6, Kemeny beats RRF ($p < 0.05$) on 8 out of 10 comparisons by a mean of 0.23 points; on average, RRF reaches only 93.5% of the boost that Kemeny does. Its only outperformance on DL19 possibly results from it being suited for information retrieval, its field of origin, but may also be statistical noise. Overall, these results further support our decision to select Kemeny ranking for the aggregation step.

5 Related Work

The holistic direction of our work is in enhancing the ranking ability of large language models. Most closely, contrast-consistent ranking (Stoehr et al., 2023) proposes to train order-enforcing probes on the latent vectors of large language models for improving rank consistency. We differentiate our method by not presuming access to model internals, which is becoming increasingly common with closed source but academically interesting LLMs such as GPT-4.

The specific empirical tasks in this paper have also seen recent progress. For passage ranking using language models, BERT-based (Devlin et al., 2019; Nogueira et al., 2020) and T5-tuned (Zhuang et al., 2023; Raffel et al., 2020) approaches represent the earliest language models for passage ranking. RankGPT (Sun et al., 2023) spearheaded much of the post-ChatGPT work, beating the supervised state of the art with an *unsupervised* LLM for the first time. Concurrently, LRL (Ma et al., 2023) reached the same conclusions using a similar method on GPT-3. Along a non-listwise direction, PRP (Qin et al., 2023) represents a pairwise method leveraging open-source large language models, as reported in Table 4.

Our secondary sorting tasks for LLMs, while less practical, have had attention as well, mostly in the context of evaluation, with BigBench (Suzgun et al., 2022) providing more than 200 distinct tasks, including one in alphabetical ordering,[1] which we enlarge and expand on in WordSort. Stoehr et al. (2023) also constructed synthetic sorting datasets for evaluating listwise ranking, but they are private and hence uncomparable.

We are not the first to establish positional biases in LLMs in general. Lu et al. (2022) are among the earliest to relate prompt order to the quality of in-context learning. Recently, Liu et al. (2023) and Wang et al. (2023a) characterized positional bias in the context of list-oriented tasks, such as question answering and response evaluation. However, we are to our knowledge the first to characterize the position biases of passage-ranking LLMs with respect to pairwise item positions.

Lastly, our paper is connected to all the meta-algorithms for improving LLM generation. As a pertinent example, Lu et al. (2022) study prompt order on in-context learning classification tasks, proposing an entropy-based statistic over development sets to find performant permutations. Aggarwal et al. (2023) make self-consistency more efficient, halting the procedure when enough samples have been collected. To keep our method in its simplest form, as self-consistency had not been applied to listwise ranking to begin with, we based our design on the original (Wang et al., 2023b).

[1] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/word_sorting

6 Conclusions and Future Work

In the present work, we introduce permutation self-consistency, a novel decoding method to improve the ranking ability of black-box LLMs by mitigating potential sensitivities and biases to list item order. We intervene on prompt list order to produce multiple rankings then return an aggregated statistic as the prediction, which intuitively has less association with the controlled variable, prompt list order. Theoretically, we prove the robustness of our method to arbitrary, fixed noise distributions under certain conditions. Empirically, our method consistently improves upon ordinary decoding on all 15 of our sorting model–dataset combinations and 13 out of 16 of our passage reranking ones. Further analyses indicate the positional biases in the reordering process of input rankings. Finally, our sensitivity analyses justify our design choices of 20 output rankings, zero sampling temperature, and the Kemeny ranking method.

In the future, permutation self-consistency can plausibly be applied to any list-oriented task, regardless of whether the underlying LLM is openly available. Examples include using LLMs for evaluation (Wang et al., 2023a) and annotating human-feedback judgments with LLMs. Another future step is to relax or reformulate our method to be differentiable, enabling training-time application in, say, RankVicuna (Pradeep et al., 2023).

Limitations

We share the same limitations as those of the original self-consistency paper (Wang et al., 2023b). We use multiple LLM calls, potentially to a commercial LLM, which would raise financial cost. Thus, practical applications may require careful weighing of quality gain against elevated expense. Nevertheless, a few calls already help, and returns rapidly diminish past 5–10 calls. We note that our method does not in practice increase latency by much, since all calls can be parallelized, and aggregation time does not rise with the number of samples.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning with LLMs. *arXiv:2305.11860*.

Alnur Ali and Marina Meilă. 2012. Experiments with Kemeny ranking: What works when? *Mathematical Social Sciences*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv:1611.09268*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality.

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

Vincent Conitzer, Andrew Davenport, and Jayant Kalagnanam. 2006. Improved bounds for computing Kemeny rankings. In *Proceedings of the 21st National Conference on Artificial Intelligence (Volume 1)*.

Gordon V. Cormack, Charles Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv:2109.10086*.

John G. Kemeny. 1959. Mathematics without numbers. *Daedalus*.

Maurice George Kendall. 1948. Rank correlation methods.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv:2307.03172*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. 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)*.

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

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankVicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv:2309.15088*.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv:2306.17563*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*.

Niklas Stoehr, Pengxiang Cheng, Jing Wang, Daniel Preotiuc-Pietro, and Rajarshi Bhowmik. 2023. Unsupervised contrast-consistent ranking with language models. *arXiv:2309.06991*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? Investigating large language models as re-ranking agent. *arXiv:2304.09542*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, et al. 2022. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. *arXiv:2210.09261*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *arXiv:2305.17926*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv:2303.18223*.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. RankT5: Fine-tuning T5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*.