Beyond Sampling: Self-Sorting for Long-Context Ranking

Juseon-Do[1†], **Sungwoo Han**[1†], *__Jingun Kwon__*[1],
Hidetaka Kamigaito[2], **Katsuhiko Hayashi**[3] **and Taro Watanabe**[2]
[1]Chungnam National University, [2]Nara Institute of Science and Technology (NAIST),
[3]The University of Tokyo
{doju00, 77sungwhan}@o.cnu.ac.kr
jingun.kwon@cnu.ac.kr
kamigaito.h@is.naist.jp
katsuhiko-hayashi@g.ecc.u-tokyo.ac.jp
taro@is.naist.jp

Abstract

Sorting is a fundamental problem in various ranking tasks. While large language models (LLMs) can sort multiple items by leveraging their ability to handle long contexts, their behavior becomes inefficient or unstable as the number of items increases, often producing inconsistent outputs. To address this issue, we propose Self-Sorting (SS) prompting. SS first generates multiple orders of items (within-list), and then generates multiple orders across these item lists (cross-list). By aggregating the rank of each item within the item lists and the rank of each list within the order lists, SS assigns a score to each item, enabling stable and consistent sorting. Experiments on five benchmark datasets for ranking tasks demonstrate the effectiveness of SS.

1 Introduction

Ranking is a fundamental component in a wide range of AI applications, including recommendation, search, passage retrieval, and knowledge graph completion. In recommendation, ranking methods order candidate items to improve personalization and user engagement (Li et al., 2021; Bi et al., 2022; Wang et al., 2024b; Lyu et al., 2024; Cao et al., 2024). In search and passage retrieval, models rank documents or passages by relevance, which has long been a focus of the IR community (Min et al., 2021; Ren et al., 2021; Drozdov et al., 2023; Sun et al., 2023). Knowledge graph completion is similarly cast as a ranking task, in which entities are ordered by their likelihood to complete missing relations (Lovelace et al., 2021; Lovelace and Rosé, 2022; Wang et al., 2023b). Because these settings often require joint processing of many candidates, ranking naturally entails long-context reasoning.

While traditional approaches rely on supervised objectives with carefully designed pointwise or pairwise losses (Burges et al., 2005, 2006; Cao et al., 2006; Liu et al., 2008; Rendle et al., 2012), recent progress in LLMs has introduced a new paradigm that treats ranking as a zero-shot reasoning problem (Sun et al., 2023; Zhuang et al., 2024b; Liu et al., 2025). By leveraging long-context capabilities, LLMs can directly process a set of candidates and produce a re-ranked list without task-specific supervision. Previous work has considered pointwise (Sachan et al., 2023), pairwise (Qin et al., 2024), and listwise methods (Adeyemi et al., 2024; Zhuang et al., 2024c) such as overlapping sliding windows (Sun et al., 2023), which are computationally heavy (Zhuang et al., 2024a). Recent work shows that listwise prompting with the full candidate set can reduce overhead while maintaining quality (Liu et al., 2025). Nonetheless, rankings remain unstable in long contexts.

Meanwhile, test-time scaling methods such as chain-of-thought (CoT) have achieved significant gains across domains (Wei et al., 2022; Yao et al., 2023; Trivedi et al., 2023; Diao et al., 2024). Nevertheless, large language models (LLMs) still struggle in long-context settings (Wang et al., 2023a; Chen et al., 2023; Wang et al., 2024a; Liu et al., 2024). Selection-based approaches, such as self-consistency (SC) and universal self-consistency (USC), aim to mitigate this by sampling multiple responses and selecting a final answer based on the model's own signals (Wang et al., 2023a; Chen et al., 2023).

However, LLMs remain inconsistent in long-context scenarios such as ranking tasks even with selection-based methods (Chen et al., 2023). We show that sampling alone does not stabilize rankings because consistency decomposes into within-list order and cross-list preference. A single stochastic process cannot align both. To address this issue, we propose Self-Sorting (SS), which in-

* corresponding author
† Equal Contribution

Findings of the Association for Computational Linguistics: EACL 2026, pages 4901–4910

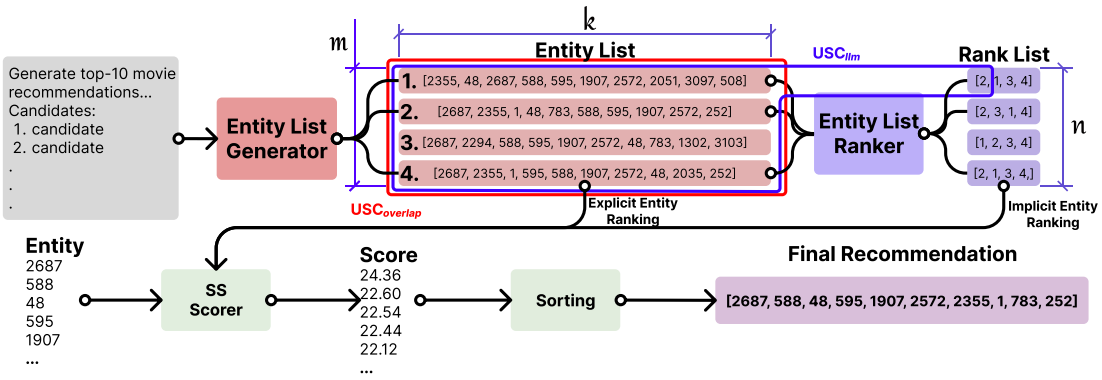

Figure 1: Overview of the SS pipeline compared with USC variants. Given a query, an LLM generates candidate entity lists. USC variants select one list. SS performs selection-time re-ranking and aggregates within-list positions (explicit) and cross-list preferences (implicit) to score entities and return the top-k set.

tegrates explicit within-list order and implicit cross-list preferences to produce a top-k set of entities. SS performs m generations and n selection-time re-rankings.

We conduct experiments on five standard benchmarks for ranking-oriented tasks: MovieLens-1M, WN18RR, FB15K-237, TREC-DL19, and Novel-Eval. Experimental results and our in-depth analysis show that SS consistently yields significant improvements over sampling-based methods across all datasets. The code will be available at https://github.com/JuseonDo/Self-Sorting.

2 Preliminary

In this section, we introduce selection-based approaches, in particular, self-consistency (SC) and universal self-consistency (USC).

SC enhances performance by sampling multiple reasoning paths and aggregating their outputs through majority voting (Wang et al., 2023a). However, SC is applicable only when exact string matching is feasible. USC extends SC to settings where exact string matching is infeasible, such as entity-list outputs (Chen et al., 2023).

Because vanilla USC is not directly applicable to the order-sensitive nature of list-ranking, we propose extending USC into two baselines. Given m candidate lists, our selector performs list-level selection to choose a single output using one of two criteria: (1) USC$_{overlap}$, which selects the list whose entities have the greatest overlap with the others, and (2) USC$_{llm}$, which uses an LLM judge to assess semantic consistency when overlap is insufficient. However, both criteria operate at the list level and typically do not model fine-grained structure such as within-list order or cross-list interactions. These limitations matter in long-context ranking, where relative ordering across multiple candidate lists is critical.

Algorithm 1 Self-Sorting

Require: the length of entity list: k, generated entity list sets: $\mathcal{L} = [\ell_1, \ldots, \ell_m]$, # of rankings: n, weight: $\lambda \in [0, 1]$
Ensure: selected entities: *selected*

function SS($\mathcal{L}, n, \lambda$)
  $R \leftarrow [\,]$
  for $i \leftarrow 1$ to n **do**
    # $R^{(i)} = [\ell_{i_1}, \ldots, \ell_{i_m}]$, from best to worst
    $R^{(i)} \leftarrow \text{LLM}_{\text{ranker}}(\mathcal{L})$
    R.add($R^{(i)}$)
  end for
  $S \leftarrow$ Dictionary(default $= 0$) # entity $\rightarrow$ score
  for $i \leftarrow 1$ to n **do**
    for $r \leftarrow 1$ to m **do**
      for $p \leftarrow 1$ to k **do**
        $e \leftarrow R^{(i)}[r][p]$
        $S[e] \leftarrow S[e] + \left(\frac{1}{r}\right)^{\lambda} \cdot \left(\frac{1}{p}\right)^{1-\lambda}$
      end for
    end for
  end for
  selected $= \arg\max_k S$
  return *selected*
end function

3 Self-Sorting (SS)

To address these limitations, we additionally propose self-sorting (SS). SS combines within-list positional evidence with cross-list preferences obtained via selection-time re-ranking, yielding a top-k set. Figure 1 shows the pipeline.

Problem Setup. Given a query q and a set of candidates, an LLM first generates m entity lists $\mathcal{L} = \{\ell_1, \ldots, \ell_m\}$, where each $\ell_j = [e_{j,1}, \ldots, e_{j,|\ell_j|}]$ is an ordered list. A listwise LLM ranker is then invoked n times to produce selection-time rankings over the m lists. Let $R^{(i)} = [\ell_{i_1}, \ldots, \ell_{i_m}]$

Method	nDCG@1	nDCG@5	nDCG@10
Random	46.51	47.39	42.05
$USC_{overlap}$	**62.02**	56.83	51.33
USC_{llm}	56.98	53.85	48.50
USC_{llm} w/ sampling	58.91	54.38	49.49
SS	**62.02**	**57.96**†	**55.24***

Table 1: Performance of GPT-4o on TREC in a long-context setting with 1,000 candidates. The best results among each part are marked in bold respectively. * and † denote statistically significant (*: $p<0.01$, †: $p<0.05$) improvements, compared to the underlined scores (typically the best baseline) on each dataset. We used paired bootstrap resampling with 100,000 random samples (Koehn, 2004) for the significance test.

denote the i-th re-ranking; the list at rank r is $R^{(i)}[r]$, and the entity at position p in that list is $R^{(i)}[r][p]$. Algorithm 1 describes the SS procedure, which aggregates explicit and implicit signals to score entities.

Explicit Signals (within-list order). Earlier positions in ℓ_j indicate greater salience among entities in that list. We treat the position p as an explicit cue and down-weight later positions.

Implicit Signals (cross-list preferences). Across lists, higher ranks under selection-time re-rankings indicate stronger global preference. We treat the list rank r in $R^{(i)}$ as an implicit cue and down-weight lower-ranked lists.

Scoring and Aggregation. We combine explicit and implicit signals to score entities and return a top-k set. For each occurrence of $e = R^{(i)}[r][p]$, the entity at position p in the list ranked r in $R^{(i)}$, we update its score as specified on line 13 of Algorithm 1. The parameter $\lambda \in [0, 1]$ balances implicit and explicit contributions. Intuitively, entities earlier within a list (small p) receive higher explicit weight, and entities appearing in lists consistently ranked higher across multiple re-rankings (small r) receive stronger implicit support. Finally, we sort entities by their aggregated scores S and output the top-k. Setting $\lambda = 0$ yields a purely explicit variant, while $\lambda = 1$ relies solely on implicit preferences.

4 Experiments

4.1 Experimental Settings

Dataset and Metrics. We evaluated our method on five benchmark datasets for ranking-oriented tasks: **MovieLens-1M (ML)** (Harper and Konstan, 2015) is a recommendation dataset with 1,000,209 ratings from 6,040 users on 3,883 movies. **WN18RR (WN)** (Dettmers et al., 2017) targets knowledge graph completion, which contains 93,003 triples over 40,943 entities and 11 relations. **FB15K-237 (FB)** (Toutanova and Chen, 2015) is a widely used benchmark for knowledge graph completion and link prediction, whose test set includes 20,466 triples spanning 10,348 entities and 224 relations. **TREC-DL19 (TREC)** (Craswell et al., 2020) is an information retrieval dataset with 43 queries and graded relevance annotations. **NovelEval (NE)** (Sun et al., 2023) evaluates retrieval models on previously unseen knowledge, consisting of 21 queries and 420 passages with graded relevance annotations across multiple domains. We used nDCG@{1, 5,10} as the evaluation metric.

Implementation Details. We considered two experimental settings: a long-context setting and a subset setting. In the long-context setting, we evaluated our method on TREC using the full set of 1,000 candidate passages per query, which includes 10 relevant passages.

In the subset setting, we evaluated all datasets, including TREC, using candidate pools of size 100, each containing 10 positive items. For TREC, we randomly sampled 100 candidates from the original pool of 1,000 passages, ensuring that all 10 relevant passages were included. For ML, WN, and FB, we sampled 500 instances, using 100 for validation and 400 for testing. In ML, for each user we treated the most recent 10 interactions as positives and sampled 90 unrelated items at random as negatives. In WN and FB, we focused on tail prediction: for each query, we ranked 100 candidate entities including the gold answer, treated the top-10 as positives, and used the remaining candidates as negatives. We generated these 100 candidates using models trained on WN and FB by SimKGC [1] with its default hyperparameters. For NE, we added 80 additional negatives to construct candidate sets of size 100.

We employed GPT-4o (OpenAI et al., 2024) and Llama3.3-70B-Instruct (Grattafiori et al., 2024) as backbone models and set top-p to 0.1 and temperature to 0.7 for both. We set $m = 8$ and $n = 8$ for entity-list generation and selection-time re-ranking. The parameter λ was chosen based on validation results for ML, WN, and FB. Since the validation split of TREC is unlabeled and NE provides no validation split, and these datasets contain only 43 and 21 queries respectively, we report results using the

[1] https://github.com/intfloat/SimKGC

Model	Strategy	ML			WN			FB			TREC			NE		
		nDCG@1	@5	@10	nDCG@1	@5	@10	nDCG@1	@5	@10	nDCG@1	@5	@10	nDCG@1	@5	@10
GPT-4o	Oracle_{list}	76.25	63.62	52.80	78.00	59.87	51.43	81.00	62.89	53.99	89.92	86.84	76.52	92.86	90.36	92.35
	Oracle_{entity}	99.75	96.95	77.83	100.00	95.31	70.54	97.00	89.40	66.99	93.02	89.71	78.59	100.00	99.69	98.97
	Random	42.50	39.88	37.01	53.06	46.59	42.33	66.49	54.16	48.29	83.33	81.45	68.02	78.57	75.72	79.43
	$\text{USC}_{overlap}$	45.00	42.14	40.26	54.02	48.67	44.07	68.97	55.82	49.54	**87.98**	84.77	74.33	83.33	79.94	84.17
	USC_{llm}	43.50	41.77	38.88	53.98	47.36	42.68	65.89	54.21	47.78	86.05	82.90	70.22	83.33	79.91	83.60
	SS	**49.00**†	**46.72***	**42.25***	**57.24**†	**49.96**†	**44.80**†	**69.03**	**57.01***	**49.86**	**87.98**	**84.92**	**75.18***	**92.86**	**88.74***	**89.58**†
Llama3.3-70B-Instruct	Oracle_{list}	55.00	45.61	37.98	74.25	52.67	42.90	77.50	60.94	51.42	87.60	86.53	74.35	90.48	79.24	80.61
	Oracle_{entity}	94.00	82.92	63.01	100.00	92.33	65.34	98.75	89.13	65.15	93.02	89.59	77.42	100.00	98.70	95.32
	Random	29.50	26.79	24.21	40.07	32.25	28.23	62.68	50.96	44.24	76.74	69.50	57.30	57.14	52.47	53.11
	$\text{USC}_{overlap}$	29.50	26.33	25.06	46.01	36.75	32.13	61.94	51.53	45.67	73.64	71.38	61.74	61.90	59.09	65.20
	USC_{llm}	31.25	28.65	25.74	46.44	37.64	33.03	62.10	49.19	42.98	87.60	81.59	69.47	64.29	58.76	62.86
	SS	**37.50***	**33.10***	**28.75***	**51.11***	**41.26***	**35.28***	**65.00**	**53.25***	**46.39**†	**88.37**	**82.75**	**72.52***	**66.67**	**65.97**	**69.41**

Table 2: Experimental results based on GPT-4o and Llama3.3-70B-Instruct.

λ that maximizes the average nDCG@{1, 5, 10}. Performance across a grid of λ is provided in Appendix A.

Compared Methods. The baselines were as follows: **Random**, which randomly chooses one from the m generated entity lists; **USC**, which selects a single list based on consistency among the m generated lists (Chen et al., 2023). We consider two USC variants: $\text{USC}_{overlap}$, which chooses the list with the greatest overlap with the others; and USC_{llm}, which uses an LLM judge to assess semantic consistency. Our method is **SS**. We also report two oracle upper bounds: Oracle_{list}, which assumes access to the best single entity list; and Oracle_{entity}, which assumes access to the best set of entities aggregated across all candidate lists.

4.2 Results

Long-Context Setting. To evaluate the effectiveness of SS in long-context settings, we conducted experiments on TREC using GPT-4o. Table 1 shows the results. SS significantly outperformed the baseline methods in the long-context scenario. These experiments demonstrate that, in the long-context scenarios, it is significantly important to consider both explicit and implicit consistency signals.

Subset Setting. Table 2 shows the results in the subset setting using GPT-4o and Llama3.3-70B-Instruct. SS consistently outperforms USC variants in nDCG when evaluated with both models. Gains are statistically significant in most settings (p<0.01 or p<0.05). These results demonstrate that, in long-context scenarios, it is crucial to consider both explicit and implicit consistency signals. Additional experimental results are in Appendix B.

4.3 Analysis

Ablation Study. We conducted an ablation study to assess the contribution of implicit signals. **USC_{llm} with sampling** uses an LLM to perform multiple re-rankings of the generated entity lists and selects the answer via majority voting. **$\text{SS}_{AvgRank}$** selects the list with the lowest average rank across selection-time re-rankings. For example, in Figure 1, the second list achieves an average rank of 1.25, which is the lowest; therefore, $\text{SS}_{AvgRank}$ recommends the second list. **SS without implicit** ($\lambda = 0.0$) leverages only explicit signals, whereas **SS without explicit** ($\lambda = 1.0$) leverages only implicit signals.

Method	GPT-4o		Llama		Latency (Llama)	
	ML	WN	ML	WN	ML	WN
Window	37.8	43.4	13.1	21.7	56h 23m	86h 54m
Random	37.0	42.3	24.2	28.2	7h 0m	6h 15m
$\text{USC}_{overlap}$	40.3	44.1	25.1	32.1	7h 0m	6h 15m
USC_{llm}	38.9	42.7	25.7	33.0	16h 21m	16h 59m
USC_{llm} w/ sampling	39.5	42.4	26.8	33.8	16h 21m	16h 59m
SS	**42.3***	**44.8***	**28.8***	**35.3***	16h 21m	16h 59m
$\text{SS}_{AvgRank}$	40.1	42.3	26.3	33.7	16h 21m	16h 59m
SS w/o implicit	40.6	44.5	26.5	34.2	16h 21m	16h 59m
SS w/o explicit	42.2	43.9	**28.8**	34.5	16h 21m	16h 59m

Table 3: Ablation study on ML and WN.

Method	Time with batching	Time without batching
Window	$\left(\lfloor \frac{c-w}{s} \rfloor + 1\right) \cdot \text{Time}_{model}$	$\left(\lfloor \frac{c-w}{s} \rfloor + 1\right) \cdot \text{Time}_{model}$
$\text{USC}_{overlap}$	$\text{Time}_{model} + C$	$m \cdot \text{Time}_{model} + C$
USC_{llm}	$2 \cdot \text{Time}_{model}$	$(m+1) \cdot \text{Time}_{model}$
SS	$2 \cdot \text{Time}_{model} + C$	$(m+n) \cdot \text{Time}_{model} + C$

Table 4: Inference time complexity. Let c denote the number of candidates, w the window size, and s the stride.

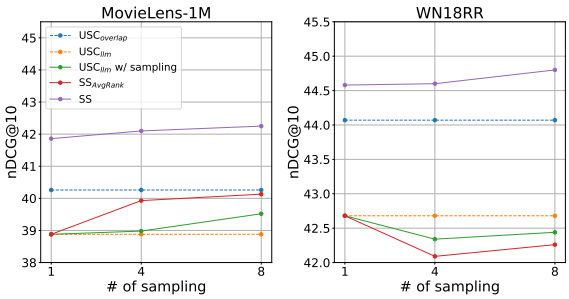

Figure 2: Effect of the number of samples on ranking performance (nDCG@10).

Dataset	Model	Method	nDCG@1	nDCG@5	nDCG@10
ML	GPT-4o	SS_{SUM}	**50.3**	**46.8**	42.2
		SS_{LOG}	49.5	46.7	42.2
		SS	49.0	46.7	**42.3**
	Llama3.3 -70B-Instruct	SS_{SUM}	39.3	**33.7**	**28.8**
		SS_{LOG}	**40.0**	33.4	28.8
		SS	37.5	33.1	28.8
WN	GPT-4o	SS_{SUM}	57.8	49.8	44.6
		SS_{LOG}	**58.0**	**50.3**	**44.9**
		SS	57.2	50.0	44.8
	Llama3.3 -70B-Instruct	SS_{SUM}	50.2	40.5	34.8
		SS_{LOG}	48.4	39.9	34.4
		SS	**51.1**	**41.3**	**35.3**

Table 5: Performance on ML and WN using SS, SS_{LOG}, and SS_{SUM}.

We also evaluated the **Window** baseline (Sun et al., 2023; Liu et al., 2025), a sliding-window approach used in long-context scenarios for ranking tasks. We also measured the latency of each method based on Llama3.3-70B-Instruct. Table 3 shows the results. Applying sampling to USC_{llm} does not effectively mitigate inconsistency in long-context scenarios. Thus, sampling alone does not stabilize rankings. In contrast, SS consistently outperforms baselines across datasets and models. These results indicate that both explicit and implicit signals are necessary for consistency-based selection in long-context ranking. Furthermore, latency measurements and Table 4 show that SS is more efficient than Window and comparable to USC_{llm}. Detailed derivations of $Time_{model}$ are provided in Appendix C.

Effect of the Number of Samples. We investigated the effect of varying the number of samples on ML and WN. Figure 2 shows the results. Increasing the number of samples for USC variants does not stabilize ranking performance. In contrast, SS consistently improves performance. As the number of samples increases, SS captures consistency more effectively, highlighting the importance of both explicit and implicit signals.

Other Scoring Functions. We also evaluated alternative variants of the scoring function on line 13 of Algorithm 1. Specifically, we tested a summation-based form ($\mathbf{SS_{SUM}}$) of $S[e] = S[e] + r^{\lambda} + p^{1-\lambda}$ and a log-weighted form ($\mathbf{SS_{LOG}}$) of $S[e] = S[e] + \lambda/\log\big(r^{-1}+1\big) + (1-\lambda)/\log\big(p^{-1}+1\big)$. Table 5 shows the results. Incorporating both explicit and implicit signals consistently improves ranking performance, regardless of the scoring form. We also provide a case study in Appendix D.

5 Conclusion

We showed that both explicit and implicit signals are essential for selection-based methods in long-context ranking. To this end, we proposed SS, which integrates explicit within-list order with implicit cross-list preferences via selection-time re-ranking. SS produces a top-k set of entities and achieves significant improvements over strong selection-based baselines across five benchmarks.

6 Limitations

We focus on improving consistency in long-context ranking by fusing explicit (within-list) and implicit (cross-list) signals at selection time. While SS is effective across benchmarks, several limitations remain.

While we selected hyperparameters λ based on validation results, assessing transfer to other datasets requires further evaluation. In addition, selection quality depends on the listwise ranker and prompt template; although we evaluated SS with both closed (GPT-4o) and open (Llama3.3-70B-Instruct) backbones, broader prompt and model variations are left for future work. We will extend our work by considering retrieval-augmented generation.

References

Mofetoluwa Adeyemi, Akintunde Oladipo, Ronak Pradeep, and Jimmy Lin. 2024. Zero-shot cross-lingual reranking with large language models for low-resource languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–656, Bangkok, Thailand. Association for Computational Linguistics.

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. MTRec: Multi-task learning over BERT for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, Dublin, Ireland. Association for Computational Linguistics.

Chris J.C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. Technical Report MSR-TR-2005-06.

Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. 2006. Adapting ranking svm to document retrieval. In *SIGIR '06 Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM.

Yuwei Cao, Nikhil Mehta, Xinyang Yi, Raghunandan Hulikal Keshavan, Lukasz Heldt, Lichan Hong, Ed Chi, and Maheswaran Sathiamoorthy. 2024. Aligning large language models with recommendation knowledge. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1051–1066, Mexico City, Mexico. Association for Computational Linguistics.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *Preprint*, arXiv:2311.17311.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2019 deep learning track. In *Text REtrieval Conference (TREC)*. TREC.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2024. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1330–1350, Bangkok, Thailand. Association for Computational Linguistics.

Andrew Drozdov, Honglei Zhuang, Zhuyun Dai, Zhen Qin, Razieh Rahimi, Xuanhui Wang, Dana Alon, Mohit Iyyer, Andrew McCallum, Donald Metzler, and Kai Hui. 2023. PaRaDe: Passage ranking using demonstrations with LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14242–14252, Singapore. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

F Maxwell Harper and Joseph A Konstan. 2015. The MovieLens datasets: history and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized transformer for explainable recommendation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4947–4957, Online. Association for Computational Linguistics.

Tie-Yan Liu, Tao Qin, and Hang Li. 2008. Query-level stability and generalization in learning to rank. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 512–519. ACM.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025. Sliding windows are not the end: Exploring full ranking with long-context large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 162–176, Vienna, Austria. Association for Computational Linguistics.

Xiang Liu, Peijie Dong, Xuming Hu, and Xiaowen Chu. 2024. LongGenBench: Long-context generation benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 865–883, Miami, Florida, USA. Association for Computational Linguistics.

Justin Lovelace, Denis Newman-Griffis, Shikhar Vashishth, Jill Fain Lehman, and Carolyn Rosé. 2021. Robust knowledge graph completion with stacked convolutions and a student re-ranking network. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1016–1029, Online. Association for Computational Linguistics.

Justin Lovelace and Carolyn Rosé. 2022. A framework for adapting pre-trained language models to knowledge graph completion. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5937–5955, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Chris Leung, Jiajie Tang, and Jiebo Luo. 2024. LLM-rec: Personalized recommendation via prompting large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 583–612, Mexico City, Mexico. Association for Computational Linguistics.

Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. 2021. Joint passage ranking for diverse multi-answer retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6997–7008, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec

Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. *Preprint*, arXiv:2306.17563.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. Bpr: Bayesian personalized ranking from implicit feedback. *Preprint*, arXiv:1205.2618.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2023. Improving passage retrieval with zero-shot question generation. *Preprint*, arXiv:2204.07496.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Minzheng Wang, Longze Chen, Fu Cheng, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, Yunshui Li, Min Yang, Fei Huang, and Yongbin Li. 2024a. Leave no document behind: Benchmarking long-context LLMs with extended multi-doc QA. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5627–5646, Miami, Florida, USA. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Yanbin Lu, Xiaojiang Huang, and Yingzhen Yang. 2024b. RecMind: Large language model powered agent for recommendation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4351–4364, Mexico City, Mexico. Association for Computational Linguistics.

Yun Cheng Wang, Xiou Ge, Bin Wang, and C.-C. Jay Kuo. 2023b. GreenKGC: A lightweight knowledge graph completion method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10596–10613, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

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. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2024a. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. *Preprint*, arXiv:2310.14122.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024b. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024, page 38–47. ACM.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024c. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 38–47, New York, NY, USA. Association for Computing Machinery.

A λ Selection

The validation results for SS on ML, WN, and FB are shown in Table 6. Based on these results, we set $\lambda = 0.9$ for ML, $\lambda = 0.7$ for WN, and $\lambda = 0.3$ for FB (for both backbones). Tables 7 and 8 present the full results for TREC and NE.

Model	λ	ML	WN	FB
GPT-4o	0.0	37.84	42.65	49.89
	0.1	37.92	42.88	49.78
	0.3	38.50	42.99	**49.93**
	0.5	39.02	42.88	49.62
	0.7	39.46	**43.27**	49.72
	0.9	**40.34**	43.03	49.50
	1.0	40.06	41.71	49.31
Llama3.3 -70B-Instruct	0.0	28.81	33.15	45.97
	0.1	28.76	33.48	45.91
	0.3	29.75	33.54	**46.36**
	0.5	30.44	33.64	46.09
	0.7	30.59	**34.00**	46.12
	0.9	**31.25**	33.34	46.08
	1.0	31.10	32.90	45.87

Table 6: Validation results (nDCG@10) on ML, WN, and FB.

Model	λ	nDCG@1	nDCG@5	nDCG@10
GPT-4o	0.0	87.98	84.84	75.14
	0.1	**87.98**	84.84	75.13
	0.3	**87.98**	**84.92**	75.18
	0.5	**87.98**	84.78	**75.28**
	0.7	**87.98**	84.38	75.20
	0.9	86.43	84.00	74.79
	1.0	85.66	83.63	74.22
Llama3.3 -70B-Instruct	0.0	87.60	79.42	69.43
	0.1	87.60	80.37	69.93
	0.3	87.60	81.79	70.95
	0.5	87.60	82.67	71.63
	0.7	**88.37**	**82.75**	**72.52**
	0.9	**88.37**	82.23	72.15
	1.0	**88.37**	82.28	71.94

Table 7: Performance across λ on TREC.

Model	λ	nDCG@1	nDCG@5	nDCG@10
GPT-4o	0.0	83.33	82.30	85.95
	0.1	83.33	82.39	85.83
	0.3	83.33	82.18	86.07
	0.5	83.33	84.10	86.51
	0.7	88.10	85.67	88.55
	0.9	**92.86**	**88.74**	89.58
	1.0	**92.86**	88.59	**89.73**
Llama3.3 -70B-Instruct	0.0	57.14	62.30	68.88
	0.1	57.14	62.92	68.93
	0.3	59.52	64.34	**69.52**
	0.5	59.52	63.54	69.42
	0.7	**66.67**	**65.97**	69.41
	0.9	64.29	65.36	68.97
	1.0	61.90	65.56	69.34

Table 8: Performance across λ on NE.

B Effectiveness in the Single-Positive Setting (FB)

We further evaluate the effectiveness of SS in a single-positive setting. We sampled 500 instances from FB, allocating 100 for validation and 400 for testing. Focusing on tail prediction, each query contains one positive and 99 negatives. As shown in Table 9, SS consistently improves ranking performance and achieves statistically significant gains over the baselines, demonstrating robustness in the one-positive KGC.

Model	Method	nDCG@1	nDCG@5	nDCG@10
GPT-4o	Random	66.25	77.72	79.99
	USC$_{overlap}$	66.00	77.53	80.17
	USC$_{llm}$	68.00	79.62	81.80
	SS	**69.50**	**81.94***	**83.73***
Llama3.3 -70B-Instruct	Random	33.50	40.14	41.68
	USC$_{overlap}$	36.75	43.92	45.02
	USC$_{llm}$	37.00	44.04	45.51
	SS	**38.25**	**46.68***	**47.92***

Table 9: Performance on FB with a single positive candidate.

C Computational Cost Comparison

Let I denote the input length, T the output length, V the vocabulary size, and d the hidden dimension of the LLM. Each layer consists of an attention module and an MLP. The computational cost can be decomposed into two phases:

Encoding. $O(I^2d + Id^2)$

Decoding. For each decoding step $t \in \{1, \ldots, T\}$, the complexity is $O\big((I+t)d + d^2 + dV\big)$, leading to a total decoding cost of $O(TId + T^2d + Td^2 + TdV)$. Thus, the overall complexity is $\text{Time}_{model} = O(I^2d + Id^2 + TId + T^2d + Td^2 + TdV)$.

D Case Study

Figure 3 presents a user profile, its gold items, candidate movies, generated entity lists, and the final recommendation lists produced by different methods, including SS. The first block shows the example user profile and its gold items. The second block presents the candidate movies. The third block provides an example prompt for generating entity lists. The fourth block shows an example prompt for ranking them. The remaining blocks illustrate the final recommendation outputs. Bolded items denote gold answers.

Given User.

Given User
Gender: Female
Age: 35-44
Occupation: other or not specified.
History:
MovieID: 2643 | Title: Superman IV: The Quest for Peace (1987) | Genres: Action|Adventure|Sci-Fi
MovieID: 266 | Title: Legends of the Fall (1994) | Genres: Drama|Romance|War|Western
MovieID: 1269 | Title: Arsenic and Old Lace (1944) | Genres: Comedy|Mystery|Thriller
MovieID: 1682 | Title: Truman Show, The (1998) | Genres: Drama
....
Gold Relevant Items
[2761, 2568, 2805, 2723, 2581, 2394, 2485, 2724, 2701, 2987]

Given Candidates.

Candidates
Candidates:
MovieID: 83 | Title: Once Upon a Time... When We Were Colored (1995) | Genres: Drama
MovieID: 1169 | Title: American Dream (1990) | Genres: Documentary
MovieID: 1059 | Title: William Shakespeare's Romeo and Juliet (1996) | Genres: Drama|Romance
MovieID: 1499 | Title: Anaconda (1997) | Genres: Action|Adventure|Thriller
MovieID: 2055 | Title: Hot Lead and Cold Feet (1978) | Genres: Comedy|Western
MovieID: 544 | Title: Striking Distance (1993) | Genres: Action
...

Generating Entity List Prompt.

System
You are an expert movie recommendation system with a clear and logical approach.
When generating movie recommendations, follow a step-by-step method, and ensure that each step is logically explained...
User
Generate top-10 movie recommendations for the given user profile.
User profile: {User Profile}
Candidate movies: {Candidates}

Generating Ranking List Prompt.

System
You are an expert movie recommendation reranker.
Your task is to carefully read the provided candidate recommendation lists and rank them based on quality...
User
I have generated the following 8 candidate recommendation lists for the given user profile:
Profile: {User profile}
Candidates: {Candidates}
Recommendations:
List 1: [2485, 2581, 2724, 2805, 2838, 2792, 1059, 3357, 2723, 1215]
List 2: [2485, 2724, 2581, 2805, 2838, 2295, 1612, 1059, 3079, 2723]
List 3: [2485, 2724, 2581, 2805, 2838, 2792, 1059, 3357, 2723, 1215]
...

SS.

Final Recommendation
[**2485**, **2581**, **2724**, **2723**, **2805**, 2792, **2761**, **2701**, 1059, 2405]

$\text{USC}_{overlap}$.

Final Recommendation
[**2485**, **2581**, **2724**, **2805**, 2838, 2792, 1059, 3357, **2723**, 1215]

USC_{llm}.

Final Recommendation
[**2485**, **2724**, **2581**, **2805**, 2838, 2295, 1612, 1059, 3079, **2723**]

Figure 3: Case study from ML.