自己整列プロンプティング：長文脈下のランキングタスクのための整列手法

Juseon Do♠ Sungwoo Han♠ Jingun Kwon♠ 上垣外 英剛♡ 林 克彦♢,♡ 渡辺 太郎♡
♠Chungnam National University ♡奈良先端科学技術大学院大学 (NAIST) ♢東京大学
{kamigaito.h, taro.watanabe}@naist.ac.jp katsuhiko-hayashi@g.ecc.u-tokyo.ac.jp

概要

整列は様々なランキングタスクにおける基本課題である．大規模言語モデル（LLM）は長文脈の把握能力を活用することで，複数の要素を整列することができるが，要素が増大した場合の動作は非効率または不安定であり，出力が一貫しないことも多い．この問題に対処するため，本稿では**自己整列（SS; Self-Sorting）プロンプティング**を提案する．SS は要素の順序（要素リスト）を複数生成した後，それらの要素リスト間の順序（順序リスト）を複数生成する．そして，要素リスト内の要素順位と順序リスト内のリスト順位を集約して，各要素をスコアリングすることで，安定した整列を実現する．ランキングタスクに関する 5 つのベンチマークデータ実験から，SS の有効性が確認された．

1 はじめに

整列は，推薦・検索・知識グラフ補完などの様々なランキングタスクにおける基本課題である．推薦においては，候補アイテムを整列することで，ユーザの意思決定を支援する [29, 17, 4]．検索は，文書やパッセージを関連度に基づいて整列するタスクであり，情報検索分野において長年研究されてきた [20, 10, 23]．知識グラフ補完も整列として定式化され，欠損した関係を補完する尤度に基づいてエンティティが整列される [16, 28]．これらのタスクでは多数の要素を対象として整列を行うため，モデルは本質的に長文脈の把握能力が必要となる．

従来，ランキングタスクに対する整列手法としては，Pointwise や Pairwise 損失のような局所的な要素の順序関係を教師付き学習する方法が主流であった [2, 3, 5, 13, 21]．一方，近年では，LLM が発展したことに伴い，Zero-shot 推論問題としての整列手法が注目を集めている [23, 32, 14]．この手法では LLM の長文脈の把握能力を活用することで，タスク固有の教師信号を用いずに，候補の要素集合をプロンプトで直接処理して整列を行う．しかし，先行研究 [22, 19, 1, 33] では，Slide-window（**Window**）法 [23] が主流であり，これは要素集合のサイズに依存した回数のモデル推論を必要とし，表 1 に示す通り，計算効率が悪い [31]．そのため，Chain-of-Thought（CoT）に代表される **Test-time Scaling** 法 [30, 26, 9] を検討することが計算効率の観点から重要となるが，サンプリングで推論を安定化する Self-Consistency（SC）や Universal Self-Consistency（USC）といった手法 [27, 6, 15] を用いても，長文脈下のランキングタスクでは，動作が不安定となり，出力が一貫しない [6]．SC や USC ではリスト単位で処理が行われるため，リスト内の要素順位やリスト間の相互作用といった細部の情報をモデルが考慮できないことが不安定さの原因になると考えられる．

表 1 推論に関する計算量の比較：c を要素集合のサイズ，w を window サイズ，s を slide サイズとする．1 回のモデル推論時間 Time_{model} の導出は付録 A.2 に示した．

手法	バッチ処理あり	バッチ処理なし
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	$2 \cdot \text{Time}_{model} + c$	$(m+n) \cdot \text{Time}_{model} + c$

この問題に対処するため，本稿では**自己整列（SS; Self-Sorting）プロンプティング**を提案する．SS では，整列対象となる要素の順序（**要素リスト**）を複数生成した後，それら要素リスト間の順序（**順序リスト**）を複数生成する．そして，要素リスト内の要素順位（**明示的情報**）と順序リスト内のリスト順位（**暗黙的情報**）に関する情報を集約して，各要素をスコアリングすることで，安定した整列を実現する．

ランキングタスクに関する 5 つのベンチマーク

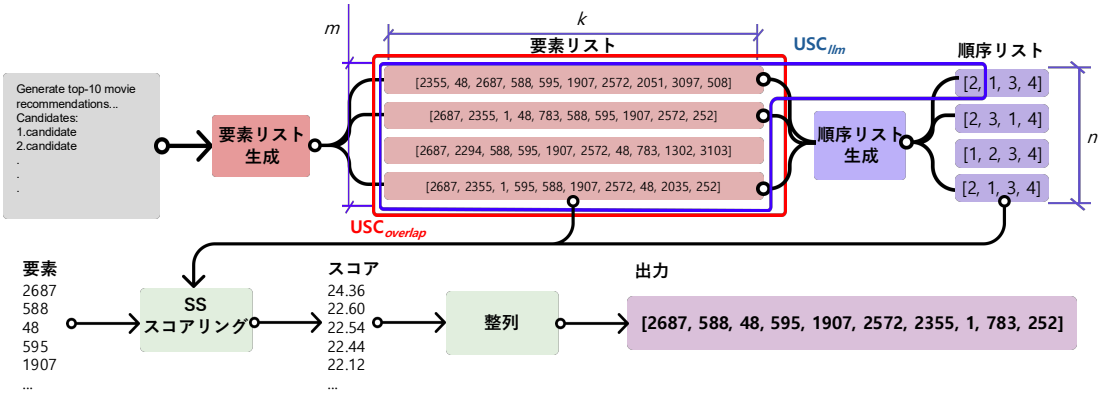

図 1　SS パイプラインと USC 系手法との比較：クエリが与えられると，LLM は要素リストを m 個生成する．USC 系手法はそのうち 1 つのリストを選択する．一方，SS は要素リスト同士の順序（順序リスト）生成を n 回行い，要素リスト内の要素順位（明示的）と順序リスト内のリスト順位（暗黙的）を集約して，要素をスコアリングし，上位 k 件を出力する．

データ MovieLens-1M，WN18RR，FB15K-237，TREC-DL19，NovelEval を用いて実験を行った．実験結果および詳細な分析から，SS はすべてのデータセットにおいて，サンプリングに基づく従来手法を一貫して大きく上回る性能を示すことが確認された．

2　前提知識

本節では，従来法となる Self-Consistency（SC）および Universal Self-Consistency（USC）について述べる．SC は，複数の推論経路をサンプリングし，それらの出力を多数決によって集約することで性能を向上させる手法である [27]．しかし，SC は完全一致による文字列マッチングが可能な場合に適用可能である．USC は，エンティティ列の出力のように完全一致が困難な設定に対して，SC を拡張した手法である [6]．本研究では USC を整列に適用可能にした 2 つのベースラインを考える．m 個の要素リストが与えられたとき，セレクタは次の 2 つの基準のいずれかに基づいて，単一のリストを選択して出力する．(1) $\text{USC}_{overlap}$ は，他のリストとの要素の重複が最大となるリストを選択する．(2) USC_{llm} は，LLM を判定器としてリストを選択する．しかし，いずれの基準もリスト単位で動作するため，リスト内の要素順位やリスト間の相互作用といった細部の情報を考慮することが難しい．長文脈下のランキングタスクにおいて，このような制限は整列の動作を不安定にし，出力の一貫性を保てないことが予想される．

3　提案法

自己整列（SS; Self-Sorting）プロンプティングは，要素リスト内の要素順位と順序リスト内のリスト順位に関する情報を組み合わせて，上位 k 件の要素を出力する．図 1 に SS のパイプラインを示す．

Algorithm 1 自己整列（Self-Sorting）

Require: 要素リストのサイズ: k, 順序リスト: $\mathcal{L} = [\ell_1, \ldots, \ell_m]$, 順序リストの数: n, 重み: $\lambda \in [0, 1]$
Ensure: selected elements: $selected$
1: **function** SS($\mathcal{L}, n, \lambda$)
2: $R \leftarrow [\,]$
3: **for** $i \leftarrow 1$ to n **do**
4: $R^{(i)} \leftarrow \text{LLM}_{\text{ranker}}(\mathcal{L})$
5: R.add($R^{(i)}$)
6: **end for**
7: $S \leftarrow$ Dictionary(default = 0) # element → score
8: **for** $i \leftarrow 1$ to n **do**
9: **for** $r \leftarrow 1$ to m **do**
10: **for** $p \leftarrow 1$ to k **do**
11: $e \leftarrow R^{(i)}[r][p]$
12: $S[e] \leftarrow S[e] + \left(\frac{1}{r}\right)^{\lambda} \cdot \left(\frac{1}{p}\right)^{1-\lambda}$
13: **end for**
14: **end for**
15: **end for**
16: $selected = \arg\max_k S$
17: **return** $selected$
18: **end function**

問題設定　クエリ q と要素集合が与えられると，LLM はまず m 個の要素リスト $\mathcal{L} = \ell_1, \ldots, \ell_m$ を生成する．ここで，各 $\ell_j = [e_{j,1}, \ldots, e_{j,|\ell_j|}]$ は順序付きリストである．その後，LLM により順序リストを n 個生成する．要素・順序リストの生成プロンプトの例は付録 A.3 に示した．

i 回目の順序リストを $R^{(i)} = [\ell_{i_1}, \ldots, \ell_{i_m}]$ と表す．順位 r に位置する要素リストは $R^{(i)}[r]$，その要素リスト内の位置 p にある要素は $R^{(i)}[r][p]$ と示す．Algorithm 1 に示した 8-15 行目の手順で，SS は要素リスト内の要素順位（明示的）と順序リスト内のリスト順位（暗黙的）を集約して要素をスコアリングする．

明示的情報（要素リスト内の要素順位）　ℓ_j にお

表 2　実験結果：各区分における最良の結果は太字で示した．* および † は，それぞれ下線付きスコア（各データセットにおける最良の比較手法）と比較して，統計的に有意な改善であることを表す（*: p<0.01，†: p<0.05）．有意性検定には，100,000 回のランダムサンプルによる対応付きブートストラップ法 [12] を用いた．

モデル	手法	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
	$\text{Oracle}_{element}$	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
	$\text{Oracle}_{element}$	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**

いて前方に位置する要素ほど，当該要素リスト内での重要度が高いことを示す．本稿では，位置 p を直接使い，後方に位置する要素の寄与を減衰させる．

暗黙的情報（順序リスト内のリスト順位）　順序リストを n 個生成した後，高い順位を得る要素リストほど，全体的な選好が強いことを示す．本研究では，$R^{(i)}$ におけるリストの順位 r を暗黙的手がかりとして，下位にランクされたリストの寄与を減衰させる．

スコアリングと集約　明示的情報と暗黙的情報を統合して要素をスコアリングし，上位 k 件を出力する．各出現 $e = R^{(i)}[r][p]$，すなわち $R^{(i)}$ において順位 r の要素リストの位置 p にある要素について，Algorithm 1 の 12 行目の式に従ってスコアを更新する．パラメータ $\lambda \in [0, 1]$ は，暗黙的寄与と明示的寄与のバランスを制御する．直感的には，リスト内で前方に位置する要素（小さい p）ほど高い明示的重みが与えられ，順序リストで上位に現れる要素リストに含まれる要素（小さい r）ほど強い暗黙的支持を受ける．最終的に，集約されたスコア S に基づいて要素を整列し，上位 k 件を出力する．

4　実験

4.1　実験設定

データセット，評価指標，および実装詳細　整列タスクに対する 5 つのデータセット，MovieLens-1M（ML），WN18RR（WN），FB15K-237（FB），TREC-DL19（TREC），NovelEval（NE）を用いて提案手法を評価した [11, 8, 25, 7, 23]．ML は推薦の標準的ベンチマークである．WN と FB は知識グラフ補完タスクであり，TREC と NE はそれぞれ検索評価のベンチマークである．評価指標として nDCG@1, 5, 10 を用いた．GPT-4o [18] と Llama3.3-70B-Instruct [24] をモデルとして使用し，いずれも top-p を 0.1，temperature を 0.7 に設定した．各データセットの詳細および設定は 付録 A.1 に示す．要素リストの生成数および順序リストの生成数は，$m = 8$，$n = 8$ に設定して実施した．パラメータ λ は，ML，WN，FB については検証データに基づいて選択した．TREC は検証分割にラベルが存在せず，NE には検証分割が存在しない．さらに，それぞれ クエリ数が 43 件および 21 件と少ないことから，nDCG@1, 5, 10 の平均を最大化する λ を用いた．

SS との比較手法　**Random** は，生成された m 個の要素リストから 1 つをランダムに選択する手法である．**USC** は，m 個の要素リスト間の一貫性に基づいて単一の要素リストを選択する手法である [6]．USC は節 2 で検討した $\text{USC}_{overlap}$ と USC_{llm} の 2 種類である．さらに，2 つの **Oracle** 上限を報告する．Oracle_{list} は，最良の要素リスト候補にアクセスできると仮定した場合の上限であり，$\text{Oracle}_{element}$ は，すべての要素リスト候補から集約した最良の要素集合にアクセスできると仮定した場合の上限である．

4.2　結果

表 2 に GPT-4o および Llama3.3-70B-Instruct による結果を示す．SS は，いずれのモデルにおいても，USC 系手法を nDCG の観点で一貫して上回る性能を示した．多くの設定において，性能向上は統計的に有意である（p<0.01 または p<0.05）．

表 3 ML および WN におけるアブレーション結果：*と下線スコアは表 2 と同じ意味である．

手法	GPT-4o		Llama		応答時間 (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

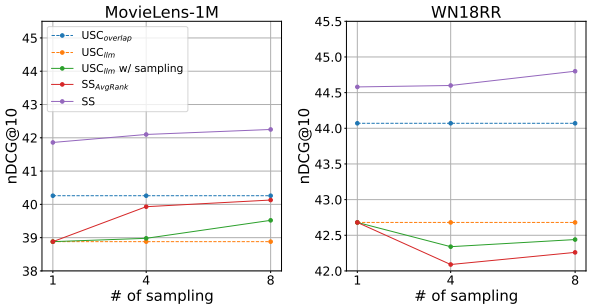

図 2 サンプル数が性能（nDCG@10）に与える影響．

4.3 分析

暗黙的情報の寄与を評価するため，アブレーションを実施した．**USC_{llm} with sampling** は，LLM を用いて生成された複数の要素リストに対して，複数回の選択を行い，結果の多数決によって最終出力するリストを 1 つ選択する手法である．**$\text{SS}_{AvgRank}$** は，n 個の順序リスト候補全てで，平均順位が最も低い要素リストを選択する手法である．例えば，図 1 において，第 2 の要素リストは平均順位 1.25 を達成しており，これが最小であるため，$\text{SS}_{AvgRank}$ は第 2 の要素リストを選択する．**SS without implicit**（$\lambda = 0.0$）は明示的情報のみを用いる手法であり，**SS without explicit**（$\lambda = 1.0$）は暗黙的情報のみを用いる手法である．また，Slide-window 法を使用する **Window** ベースライン [23, 14] も評価した．さらに，`Llama3.3-70B-Instruct` に基づき，各手法の応答時間を測定した．

結果を表 3 に示す．USC_{llm} にサンプリングを適用しても，結果の不整合は十分に緩和されなかった．このことから，サンプリングのみでは長文脈下のランキングタスクにおける整列は安定化しないことが分かる．一方で，SS はすべてのデータセットおよびモデルにおいて，一貫してベースラインを上回る性能を示した．これらの結果は，長文脈下のランキングタスクにおいて，明示的情報と暗黙的情報の双方が必要であることを示唆している．さらに，応答時間の測定結果および表 1 の計算量から，SS は Window よりも効率的であり，USC_{llm} と同程度の計算コストであることが分かる．

表 4 ML と WN における SS，SS_{LOG}，および SS_{SUM} の性能．

データ	モデル	手法	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**

サンプル数の影響 ML および WN において，サンプル数（SS では n に該当）を変化させた際の影響を調査した．結果を図 2 に示す．USC 系手法では，サンプル数を増加させても整列の性能は安定しない．一方，SS ではサンプル数の増加に伴い，一貫して性能が向上した．サンプル数が増えるにつれて，SS は一貫性をより効果的に捉えるようになり，明示的情報と暗黙的情報の双方を考慮したスコアリング関数がサンプル数に対して頑健であることが示された．

他のスコアリング関数 Algorithm 1 の 13 行目におけるスコアリング関数について，別の方法も評価した．具体的には，$S[e] = S[e] + r^{\lambda} + p^{1-\lambda}$ と定義する加算型（**SS_{SUM}**）と，$S[e] = S[e] + \lambda/\log(r^{-1}+1) + (1-\lambda)/\log(p^{-1}+1)$ と定義する対数重み付き型（**SS_{LOG}**）を検討した．結果を表 4 に示す．これらのスコアリング関数の違いに依らず，明示的情報と暗黙的情報の双方を組み込むことで，性能が一貫して向上することが確認された．

5 結論

本稿では，長文脈下のランキングタスクにおいて，明示的情報と暗黙的情報の双方を考慮することの重要性を検討し，それらを組み込んだ自己整列（SS）に基づく新たな LLM ベースの整列手法を提案した．SS は 5 つのランキングタスクのベンチマークすべてにおいて，USC などの強力な従来手法を大きく上回る性能向上を達成した．今後は，LLM ベースの整列手法という利点を活かして，解答の解釈性向上に関する手法を検討したい．

謝辞

本研究は JSPS 科研費 JP23K28148, JP24K02993 の助成を受けたものです.

参考文献

[1] M. Adeyemi, A. Oladipo, R. Pradeep, and J. Lin. Zero-shot cross-lingual reranking with large language models for low-resource languages. In L.-W. Ku, A. Martins, and V. Srikumar eds., **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**, pp. 650–656, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[2] C. J. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. Technical Report MSR-TR-2005-06, August 2005.

[3] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. Platt, and T. Hoffman eds., **Advances in Neural Information Processing Systems**, Vol. 19. MIT Press, 2006.

[4] Y. Cao, N. Mehta, X. Yi, R. Hulikal Keshavan, L. Heldt, L. Hong, E. Chi, and M. Sathiamoorthy. Aligning large language models with recommendation knowledge. In K. Duh, H. Gomez, and S. Bethard eds., **Findings of the Association for Computational Linguistics: NAACL 2024**, pp. 1051–1066, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[5] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. 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**, pp. 186–193. ACM, August 2006.

[6] X. Chen, R. Aksitov, U. Alon, J. Ren, K. Xiao, P. Yin, S. Prakash, C. Sutton, X. Wang, and D. Zhou. Universal self-consistency for large language model generation, 2023.

[7] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. Overview of the trec 2019 deep learning track. In **Text REtrieval Conference (TREC)**. TREC, March 2020.

[8] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2d knowledge graph embeddings. **CoRR**, abs/1707.01476, 2017.

[9] S. Diao, P. Wang, Y. Lin, R. Pan, X. Liu, and T. Zhang. Active prompting with chain-of-thought for large language models. In L.-W. Ku, A. Martins, and V. Srikumar eds., **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1330–1350, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[10] A. Drozdov, H. Zhuang, Z. Dai, Z. Qin, R. Rahimi, X. Wang, D. Alon, M. Iyyer, A. McCallum, D. Metzler, and K. Hui. PaRaDe: Passage ranking using demonstrations with LLMs. In H. Bouamor, J. Pino, and K. Bali eds., **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 14242–14252, Singapore, Dec. 2023. Association for Computational Linguistics.

[11] F. M. Harper and J. A. Konstan. The MovieLens datasets: history and context. **ACM Transactions on Interactive Intelligent Systems**, 5(4):19:1–19:19, Dec. 2015.

[12] P. Koehn. Statistical significance tests for machine translation evaluation. In D. Lin and D. Wu eds., **Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing**, pp. 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[13] T.-Y. Liu, T. Qin, and H. Li. Query-level stability and generalization in learning to rank. In **ICML '08: Proceedings of the 25th international conference on Machine learning**, pp. 512–519. ACM, July 2008.

[14] W. Liu, X. Ma, Y. Zhu, Z. Zhao, S. Wang, D. Yin, and Z. Dou. Sliding windows are not the end: Exploring full ranking with long-context large language models. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar eds., **Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 162–176, Vienna, Austria, July 2025. Association for Computational Linguistics.

[15] X. Liu, P. Dong, X. Hu, and X. Chu. LongGenBench: Long-context generation benchmark. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen eds., **Findings of the Association for Computational Linguistics: EMNLP 2024**, pp. 865–883, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics.

[16] J. Lovelace and C. Rosé. A framework for adapting pre-trained language models to knowledge graph completion. In Y. Goldberg, Z. Kozareva, and Y. Zhang eds., **Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing**, pp. 5937–5955, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[17] H. Lyu, S. Jiang, H. Zeng, Y. Xia, Q. Wang, S. Zhang, R. Chen, C. Leung, J. Tang, and J. Luo. LLM-rec: Personalized recommendation via prompting large language models. In K. Duh, H. Gomez, and S. Bethard eds., **Findings of the Association for Computational Linguistics: NAACL 2024**, pp. 583–612, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[18] OpenAI. Gpt-4o system card, 2024.

[19] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, L. Yan, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, and M. Bendersky. Large language models are effective text rankers with pairwise ranking prompting, 2024.

[20] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J.-R. Wen. RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih eds., **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**, pp. 2825–2835, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[21] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback, 2012.

[22] D. S. Sachan, M. Lewis, M. Joshi, A. Aghajanyan, W. tau Yih, J. Pineau, and L. Zettlemoyer. Improving passage retrieval with zero-shot question generation, 2023.

[23] W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, and Z. Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In H. Bouamor, J. Pino, and K. Bali eds., **Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing**, pp. 14918–14937, Singapore, Dec. 2023. Association for Computational Linguistics.

[24] L. Team. The llama 3 herd of models, 2024.

[25] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In A. Allauzen, E. Grefenstette, K. M. Hermann, H. Larochelle, and S. W.-t. Yih eds., **Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality**, pp. 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

[26] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In A. Rogers, J. Boyd-Graber, and N. Okazaki eds., **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics.

[27] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[28] Y. C. Wang, X. Ge, B. Wang, and C.-C. J. Kuo. GreenKGC: A lightweight knowledge graph completion method. In A. Rogers, J. Boyd-Graber, and N. Okazaki eds., **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 10596–10613, Toronto, Canada, July 2023. Association for Computational Linguistics.

[29] Y. Wang, Z. Jiang, Z. Chen, F. Yang, Y. Zhou, E. Cho, X. Fan, Y. Lu, X. Huang, and Y. Yang. RecMind: Large language model powered agent for recommendation. In K. Duh, H. Gomez, and S. Bethard eds., **Findings of the Association for Computational Linguistics: NAACL 2024**, pp. 4351–4364, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[30] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. **CoRR**, abs/2201.11903, 2022.

[31] H. Zhuang, Z. Qin, K. Hui, J. Wu, L. Yan, X. Wang, and M. Bendersky. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels, 2024.

[32] S. Zhuang, H. Zhuang, B. Koopman, and G. Zuccon. 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, p. 38–47. ACM, July 2024.

[33] S. Zhuang, H. Zhuang, B. Koopman, and G. Zuccon. 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, p. 38–47, New York, NY, USA, 2024. Association for Computing Machinery.

A　付録

A.1　データセット

MovieLens-1M（ML）は，6,040 人のユーザによる 3,883 本の映画に対する 1,000,209 件の評価から構成される推薦データセットである．**WN18RR（WN）**は知識グラフ補完を対象としたデータセットであり，40,943 個のエンティティと 11 種類の関係からなる 93,003 件のトリプルを含む．**FB15K-237（FB）**は，知識グラフ補完およびリンク予測のための広く用いられているベンチマークであり，テストセットには 10,348 個のエンティティと 224 種類の関係にまたがる 20,466 件のトリプルが含まれる．**TREC-DL19（TREC）**は，43 件のクエリと段階的関連度アノテーションを含む情報検索データセットである．**NovelEval（NE）**は，これまでに出現していない知識に対する検索モデルの評価を目的としたデータセットであり，複数ドメインにわたる段階的関連度アノテーション付きの 21 件のクエリと 420 本のパッセージから構成される．

ML，WN，FB については，500 インスタンスをサンプリングし，100 件を検証用，400 件をテスト用として使用した．ML では，各ユーザについて直近の 10 件のインタラクションを正例とし，無関係な 90 件のアイテムをランダムに負例としてサンプリングした．WN および FB では，テール予測に焦点を当て，各クエリについて正解エンティティを含む 100 件の候補エンティティを順位付けし，上位 10 件を正例，残りを負例として扱った．これら 100 件の候補は，WN および FB 上で学習された SimKGC [1] を用い，デフォルトの設定で生成した．TREC では，元の 1,000 件の候補集合から，関連する 10 本のパッセージを含む 100 件をランダムにサンプリングした．NE では，80 件の追加負例を加えることで，候補集合のサイズを 100 に統一した．

実装の詳細として，NVIDIA RTX A6000 を 4 基使用した．その他の設定はすべての手法で共通である．

A.2　モデル推論の計算時間

I を入力長，T を出力長，V を語彙サイズ，d を LLM の隠れ次元とする．各層は，アテンションモジュールと MLP から構成される．計算コストは，以下の 2 つの段階に分解できるため，$\mathrm{Time}_{model} = O(I^2d + Id^2 + TId + T^2d + Td^2 + TdV)$ となる．

Encoding　$O(I^2d + Id^2)$.

Decoding　各デコーディング時間 $t \in \{1, \ldots, T\}$ に対して，計算量は $O\big((I+t)d + d^2 + dV\big)$ であり，トータルでは $O(TId + T^2d + Td^2 + TdV)$ となる．

A.3　生成プロンプトの例

図 3 に，映画推薦を対象とした要素リストおよび順序リストの生成プロンプトを示した．User profile は対象ユーザが過去に視聴した映画や嗜好などが含まれた履歴情報で，Cadidates は推薦対象となる映画（要素）の集合である．これらは別ファイルからプロンプトに組み込まれる．

要素リストの生成プロンプト.

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}
```

順序リストの生成プロンプト.

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]
...
```

図 3　ML における映画推薦を例とした生成プロンプト.

1）https://github.com/intfloat/SimKGC