CS 6103 Project: Can LLMs Learn Reasonable Rankings Using Pairwise Comparisons?

Afrin Dange
CMInDS
IIT Bombay
22m2161@iitb.ac.in

Vaibhav Singh
CSE
IIT Bombay
22m0827@iitb.ac.in

Arpit Agarwal
Instructor, CSE
IIT Bombay
aarpit@cse.iitb.ac.in

Abstract

Sports tournaments utilize pairwise game results to learn a final ranking of teams and/or players using deterministic algorithms such as ELO or probabilistic algorithms such as TrueSkill. In this work, we examine whether LLMs can inherently model a rank aggregation algorithm similar to TrueSkill. Specifically, we study whether an LLM can generate an ordering from in-context pairwise preference data. Through experiments on ICC Test championship data and NFL season rankings, we observe that Chain of thought prompting with in-context pairwise preference data provides competitive rankings similar to classical methods like BTL, TrueSkill, Elo and Glicko and show promising capabilities in pairwise preference aggregation tasks using LLMs. However, we also find that LLMs are sensitive to the order of in-context examples, which may affect their reliability in high-stakes ranking tasks.

1 Introduction

In many applications involving decision making such as product reviews, response ranking in search engines, learning to rank in machine learning, relative preferences among items becomes an metric of choice as they are straightforward to express and interpret as compared to assigning an absolute score to each item. Morever, relative preferences are contextually sensitive like human decision making. In such a scenario, aggregation of local preferences to generate a global relative ordering among items becomes a crucial problem. Early statistical models of aggregating local pairwise comparisons have been in studied in psychology and statistics, notably by Thurstone [17] and Mosteller [12] and later by Bradley and Terry [1]. More recent bayesian models like Elo, Glicko, and TrueSkill [6, 11]) focus on dynamic skill rating based on pairwise game results. Similarly, *learning-to-rank* algorithms, involve pairwise and listwise loss formulations [4, 3] and deep neural architectures [5, 13] for generating a global ranking.

1.1 Classical Pairwise Comparison Models

Classical paired-comparison models assume a latent score μ_i for each item i. For two items i, j, the probability that i wins (or is preferred) over j is modeled as $P(i \succ j) = F(\mu_i - \mu_j)$ for some function F. Thurstone-Mosteller Model [17, 12] uses the normal CDF $F = \Phi$. BTL [1] uses the logistic function. In BTL, $P(i \succ j) = \exp(\mu_i)/(\exp(\mu_i) + \exp(\mu_j))$. Bradley–Terry Model is widely applied to ranking teams in leagues, products by consumer preferences, and aggregating pairwise votes. The desirable properties in BTL include: interpretability, convex likelihood, and consistency with probability axioms. Both Thurstone–Mosteller and Bradley–Terry models assume each match outcome is independent given individual skills and require sufficient data to converge.

Algorithm 1 Bradley-Terry MLE

function BRADLEY_TERRY_LL(θ, pref_data)
  $S \leftarrow exp(\theta)$
  $\ell(\theta; \text{pref\_data}) = 0$
  for each d in pref_data **do**
    $(w, l) \leftarrow d$
    $P(w \text{ wins}) \leftarrow \dfrac{S[w]}{S[w] + S[l]}$
    $\ell(\theta; \text{pref\_data}) \leftarrow \ell(\theta; \text{pref\_data}) + \log P(w \text{ wins})$
  end for
  Output: $-\ell(\theta; \text{pref\_data}) + \frac{\lambda}{2} \sum_{i=1}^{n} \theta_i^2$
end function

Input: pref_data
$\theta = [0, 0, ..., 0]$
minimize(bradley_terry_ll, θ, pref_data, BFGS)
$S \leftarrow exp(\theta)$
Output: S =0

Algorithm 2 Elo Rating

Input: pref_data, K, n
$S = [1000, 1000, ..., 1000]$
for $i = 1$ to num_iters **do**
  for each d in pref_data **do**
    $(w, l) \leftarrow d$
    $P(w \text{ wins}) \leftarrow \dfrac{1}{1 + 10^{(S[l] - S[w])/K}}$
    $P(l \text{ wins}) \leftarrow \dfrac{1}{1 + 10^{(S[w] - S[l])/K}}$
    $S[w] \leftarrow S[w] + K \times (1 - P(w \text{ wins}))$
    $S[l] \leftarrow S[l] - K \times P(l \text{ wins})$
  end for
end for
Output: S =0

1.2 Rank Aggregation Models

Apart from pairwise observations, one may have full or partial ranking lists from multiple sources. Rank aggregation methods consider ranking from several input permutations to generate a global ranking. *Mallows model* and *Plackett–Luce model* are two statistical models for ranking aggregation and interpretation. *Mallows model* [10] assumes rankings are noisy permutations around a central "modal" permutation, while the *Plackett–Luce model* [14, 9], defines a probability on permutations via successive choice. In Plackett–Luce, the probability of a particular ordered list $(i_1, i_2, \ldots, i_n)$ is given by,

$$P(i_1, \ldots, i_n) = \prod_{k=1}^{n} \frac{\exp(\mu_{i_k})}{\sum_{j=k}^{n} \exp(\mu_{i_j})}.$$

Plackett–Luce generalizes Bradley–Terry (BTL) to full rankings.

1.3 Bayesian and Probabilistic Rating Systems

Sports analytics and online gaming applications often use Bayesian or state-space models to maintain rankings over time. Elo rating system updates player ratings after each match assuming gaussian performance noise. Glicko extends Elo, adding uncertainty and dynamically adjusting volatility. Elo, Glicko implicitly assume a Bradley–Terry-like win probability and perform sequential Bayesian updates.

TrueSkill [7] is a modern extension developed for Xbox Live games. Each player's skill is a Gaussian random variable; teams' performances are sums of member skills. TrueSkill uses approximate message passing on factor graphs to update beliefs on team skills. Compared to Elo, TrueSkill explicitly models draw probabilities, can handle any number of players or teams per match, and tracks uncertainty (variance of each skill) to converge faster.

1.4 Large Language Models and Preference Learning

Transformer-based models have also been applied to listwise ranking, multi-turn response ranking, and cross-modal ranking. In recommender systems, graph neural networks [8] are employed to re-rank retrieved documents. While Large language models (LLMs) are not inherently pairwise rankers, they can be utilized for ranking-like tasks. For example, one can prompt an LLM to choose which of two answers is better, effectively using it as a preference comparator [15, 16]. LLMs can capture nuanced preferences if properly guided. We note that this is an emerging frontier. In this work,

we study whether an LLM can be used as a rank aggregator and outperform classical frameworks like BTL and TrueSkill.

2 Methodology

Many studies showcase that LLMs demonstrate emergent in-context learning ability [2, 18]. In-context learning enables LLMs to perform complex tasks with few-shot prompt demonstrations. In this work, we exploit the in-context learning ability of LLMs to perform global rank aggregation based on local pairwise preferences. Previous works [15, 16] utilize LLMs to generate pairwise preference confidences using ICL. These pairwise preference confidences are then aggregated using statistical models like Elo, TrueSkill and BTL.

In this work, we study whether LLMs can inherently model rank aggregation methods like Elo, TrueSkill and BTL. Specifically we look at three ICL settings:

Direct Prompt In direct prompting, we query the LLM with a single prompt, with preference data provided in-context and expect the LLM to generate the ranked response directly without any intermediate reasoning steps.

Self Consistency We extend direct prompting to self consistency prompting by querying the LLM multiple times with the same input k times. The LLM then generates k different rankings, which is then aggregated into one, using majority voting at every rank position.

Chain of Thought In CoT, we introduce explicit reasoning steps in the prompt, where we direct the LLM to generate several heuristic-based candidate rankings. The candidate rankings are then aggregated using majority voting.

Refer to Appendix A for prompt details.

3 Experiments

3.1 Implementation Details

We conduct our rank aggregation experiments with `gemini-2.0-flash` with disabled code execution and function calling using three ICL settings: (1) Direct Prompting, (2) Self Consistency, (3) Chain of Thought. We include all pairwise preference data into the prompt as `gemini-2.0-flash` supports an input context length upto $1,048,576$ tokens. We implement Elo, Glicko, TrueSkill and the Bradley–Terry algorithms as our baseline.

3.2 Evaluation Metrics, and Datasets

Ranking models are often evaluated with rank-aware metrics. Rank-aware metrics consider both the relevance of items and their positions in a ranked list. In IR and recommendation, common rank-aware metrics include Precision@k, Mean Average Precision (MAP), Discounted Cumulative Gain (DCG), Normalized DCG (NDCG), and Mean Reciprocal Rank (MRR). These metrics emphasize both the relevance of the items and their relative order. Accuracy or log-likelihood of held-out comparisons are common metrics for evaluating models built to predict 1v1 outcomes. In the case of sporting events, model performances are often measured using the prediction accuracy of game outcomes or rank correlation techniques like Kendall's τ, or Spearman's ρ between the predicted standings and the gold standings.

For our experiments, we use the 2023-2025 cycle of the ICC Test Championship tournament. The data includes 59 unique pairwise match results for 5-day test cricket matches played between 9 teams, where the number of matches played by each team may differ from the others. The data set comprises a gold ordering among the teams, which is an aggregation of the outcomes of the 1v1 match. Rank aggregation algorithms in sporting events generally consider the number of matches played by each team and their wins, losses, and draws. We also use the NFL's regular season rankings from 2018 to 2023 with 32 teams participating every year.

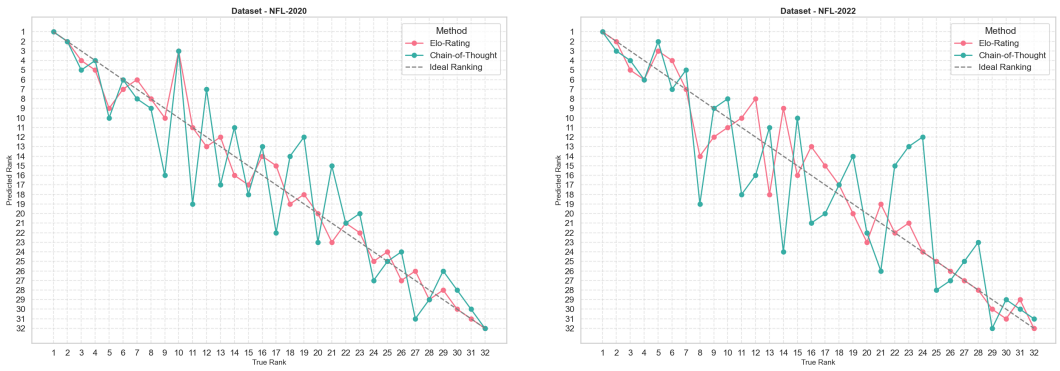

Figure 1: LLM-based methods show higher variance—particularly under different prompting strategies—highlighting their sensitivity to input format and order as compared to a more deterministic algorithm like Elo.

4 Results

4.1 Evaluate Ranking Performance

Table 1 shows the comparison between rank aggregation efficacy of classical rank aggregation algorithms against LLM-based rank aggregation with ICL on the ICC Test Championship data. We observe that the Bradley-Terry model achieves the highest overall performance across all rank-aware metrics. For NFL, in Table 2, we observe that classical methods like Elo and BTL score the best NDCG in 5 out of 6 cases. Among the ICL settings, Chain-of-thought aggregation showcases consistent improvement over direct prompting and self consistency prompting for the ICC dataset. However, direct prompting and Chain-of-thought achieve comparable performances for the NFL dataset.

4.2 Effect of Shuffling Pairwise Comparisons

As shown in Table 3 & 4, we observe that the LLM-based ranking is sensitive to the relative order of the in-context examples. We observe larger margin of difference in performance between two permutations for longer context sizes.

Table 1: Comparison of different ranking algorithms and prompting strategies on the ICC 2023–2025 dataset using various ranking-based metrics. Higher is better; best values are highlighted in green and second-best in yellow.

Metric	Elo	Glicko	TrueSkill	Bradley Terry	Direct Prompt	Self Consistency	Chain of Thought
Kendall's τ	0.611	0.667	0.611	0.778	0.667	0.667	0.778
Spearman's ρ	0.817	0.833	0.817	0.900	0.817	0.817	0.900
Top-3 Accuracy	1.000	1.000	1.000	1.000	0.667	0.667	0.667
Top-5 Accuracy	1.000	1.000	1.000	1.000	0.800	0.800	1.000
NDCG	0.928	0.928	0.922	0.957	0.948	0.948	0.955

5 Conclusion

In this work, we investigated whether large language models (LLMs) can perform global rank aggregation using only in-context pairwise comparisons. We evaluated three prompting strategies: Direct Prompting, Self-Consistency, and Chain-of-Thought against classical probabilistic ranking algorithms such as Bradley-Terry, Elo, Glicko, and TrueSkill on datasets from ICC Test Championship and NFL seasons. Our experiments show that while classical methods consistently achieve the best

Table 2: NDCG performance across ranking models and prompting strategies on NFL datasets (2018–2023). The best score per dataset is highlighted in green, and the second-best in yellow.

Dataset	Elo	Glicko	TrueSkill	Bradley-Terry	Direct Prompt	Chain of Thought
NFL-2018	0.938	0.878	0.814	0.852	0.858	0.816
NFL-2019	0.803	0.854	0.857	0.889	0.852	0.804
NFL-2020	0.896	0.890	0.857	0.912	0.846	0.897
NFL-2021	0.754	0.741	0.741	0.750	0.851	0.880
NFL-2022	0.848	0.828	0.839	0.799	0.769	0.795
NFL-2023	0.915	0.899	0.903	0.908	0.813	0.877

Table 3: Comparison of prompting strategies with and without input shuffling on the ICC dataset using correlation and accuracy metrics. We observe that the LLM is sensitive to the order of the in-context examples, which is not ideal.

Metric	Permutation 1			Permutation 2		
	Direct Prompt	Self Consistency	Chain of Thought	Direct Prompt	Self Consistency	Chain of Thought
Kendall's τ	0.667	0.667	0.778	0.667	0.667	0.833
Spearman's ρ	0.817	0.817	0.900	0.850	0.850	0.933
Top-3 Accuracy	0.667	0.667	0.667	1.000	1.000	1.000
Top-5 Accuracy	0.800	0.800	1.000	1.000	1.000	1.000
nDCG	0.948	0.948	0.955	0.924	0.924	0.957

Table 4: Comparison of Direct Prompt and Chain of Thought prompting strategies on NFL datasets (2018–2023) under Permutation 1 and Permutation 2 settings. For each method and metric, the better value is highlighted in yellow.

Dataset	Metric	Permutation 1		Permutation 2	
		Direct Prompt	Chain of Thought	Direct Prompt	Chain of Thought
NFL-2018	Kendall's τ	0.101	0.040	0.048	0.113
	NDCG	0.858	0.816	0.817	0.806
NFL-2019	Kendall's τ	0.000	-0.121	-0.060	-0.004
	NDCG	0.852	0.804	0.810	0.858
NFL-2020	Kendall's τ	0.141	0.153	-0.073	-0.129
	NDCG	0.846	0.897	0.815	0.754
NFL-2021	Kendall's τ	0.028	0.149	0.133	-0.097
	NDCG	0.851	0.880	0.885	0.751
NFL-2022	Kendall's τ	-0.121	-0.020	-0.073	-0.044
	NDCG	0.769	0.795	0.763	0.766
NFL-2023	Kendall's τ	0.056	0.105	0.121	–
	NDCG	0.813	0.877	0.780	–

performance across ranking metrics, LLMs with structured prompting, particularly, Chain of thought provides competitive rankings and show promising capabilities in pairwise preference aggregation tasks using LLMs. However, we also find that LLMs are sensitive to the order of in-context examples, which may affect their reliability in high-stakes ranking tasks.

References

[1] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel

Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Christopher J. C. Burges, Robert Ragno, and Quoc V. Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems (NIPS) 19*, pages 193–200, 2006.

[4] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, Greg Hullender, Xia Cao, Hang Shen, Yu Pandey, and Tapas Gupta. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 89–96, 2005.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186, 2019.

[6] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill: A bayesian skill rating system. In *Advances in Neural Information Processing Systems (NIPS) 19*, pages 569–576, 2006.

[7] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: A bayesian skill rating system. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[8] Zijian Li, Qingyan Guo, Jiawei Shao, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. Graph neural network enhanced retrieval for question answering of llms, 2024.

[9] R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. John Wiley & Sons, 1959.

[10] Colin L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[11] Tom Minka, Ryan Cleven, and Yordan Zaykov. Trueskill 2: An improved bayesian skill rating system. Technical Report MSR-TR-2018-8, Microsoft Research, 2018.

[12] Frederick Mosteller. Remarks on the method of paired comparisons: I. the least squares solution assuming equal standard deviations and equal correlations. *Journal of the American Statistical Association*, 46(254):936–954, 1951.

[13] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

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

[15] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[16] Vaishnavi Shrivastava, Ananya Kumar, and Percy Liang. Language models prefer what they know: Relative confidence estimation via confidence preferences, 2025.

[17] L. L. Thurstone. A law of comparative judgment. *Psychological Review*, 34(4):273–286, 1927.

[18] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently, 2023.

A Prompt Details

Direct Prompting/Self Consistency

You are given pairwise match results between teams, where each match can result in a win, loss, or a draw. Generate the final team rankings based on overall wins, losses, draws and number of matches played by a team.
Respond with the ranked list of only team names in descending order of skill in JSON format. The list should contain all teams.
Example:

```
["Team A", "Team B", "Team C"]
```

Figure 2: Direct Prompting/Self Consistency for computing team rankings from 1v1 match results.

Chain-of-Thought Prompting

1. **Create a markdown table of each team's match outcomes:**

Team	Wins	Losses	Draws	Win %	Points	Net Wins
Example	2	1	1	66.67%	2.5	1

Scoring Rules:

- Win = 1 point
- Draw = 0.5 points
- Loss = 0 points
- Win Percentage = `(Wins / (Wins + Losses + Draws)) * 100`
- Net Wins = Wins – Losses

If a team is not mentioned in any match, include them with zeroes in all columns.

2. **Generate 3 Candidate Rankings:**

Method	Ranking
1. Win Percentage	["Team A", "Team B", "Team C"]
2. Points	["Team A", "Team C", "Team B"]
3. Head-to-Head	["Team B", "Team A", "Team C"]

3. **Compare Ranking Candidates:**
   - Head-to-head accuracy: Fewer contradictions with match outcomes.
   - Win Percentage: Higher win percentage indicates better overall performance.
   - Points: Total points accumulated.
   - Win Consistency: Longest win/loss streak or least alternation.
   - Lexical Tie-breaker: Alphabetical order if still tied.

4. **Tiebreaker Protocol:**
   1. Direct match outcomes
   2. Average margin of victory (e.g., if available)
   3. Win/loss streak consistency
   4. Alphabetical order

5. **Verification Table:**

Team Pair	Actual Winner	Higher Ranked Team	Consistency
Team A vs Team B	Team A	Team A	✓

6. **Final Output (JSON):**

```
["Team A", "Team B", "Team C"]
```

7. **Input Example:**

```
Team A vs Team B, Team A won
Team B vs Team C, Team C won
Team A vs Team C, Match Drawn
```

Figure 3: Chain-of-Thought Prompting for computing team rankings from 1v1 match results.