Discovering Criteria for ranking aspects in reward modeling

Mukul Namagiri

July 19, 2025

1 Abstract

With the increased latency levels in modern large language models (LLMs) and the growing use of techniques like reward modeling, Reinforcement Learning-based approaches have become more popular for fine-tuning LLM responses. In this paper, we introduce a new method that employs aspect extraction to identify specific criteria within the generated candidate responses. These criteria are then used to evaluate responses based on their positive or negative aspects, reducing the time needed for human preference labeling. Lastly, we present an automated pipeline that evaluators can use to assess responses efficiently, minimizing downtime in the preference optimization process.

2 Introduction

An AI system that is aligned towards human preference is not just desirable but necessary. The system should be devoid of any hacking strategies and should possess a risk-averse approach in its response generation. The emerging phenomenon of intelligence remains a largely unsolved mystery to say the least. The so-called intelligence and linguistic provenance in these models come from the statistical patterns on which these models are trained rather than the explicit logical inference their human counterparts seem to show. Humans most often tend to get it from exposing themselves to different environments and situations but it is physically impossible for these models. Hence alignment of these models is far more challenging task.

The language models of the time exhibit several remarkable skills from coding, reasoning, and textual generation to going beyond natural language processing tasks. These abilities of the llms have generalized to solve complex real-world problems like autonomous agent control to robotics have made the front and center of the conversation of the AI. With the recent advancements like chain-of-thought reasoning, reflective reasoning, tree search etc have made these llms from simple auto regressive machines that simply predict the next token to cognitive machines with the ability to reason in several stages and produce the most accurate response.

Despite their remarkable capabilities llms are prone to significant shortcomings in maintaining context relevance and logical consistency for longer periods although some of the pre training methods by tuning the architectures of the models have solved this problem to some extent like capturing contextual interdependency in tasks like question answering and maintaining generative coherence in producing tokens but for enhancing the practical utility of these models post training is the way to go. The period from the later parts of the 2020s have seen significant strides in the utility of RLHF for fine tuning the responses. The post training phase of the language models have diversified to address the problems of domain specificity, ethical robustness, integration of multi modality etc recent years which led to the development of techniques like RAG (Retrieval augmented generation), different variants of preference optimization and increased adoption of techniques like mixture of experts etc. By 2024 traditional fine tuning techniques has evolved into RL based strategies.

3 Back ground

The following different aspects are presented in this section.

- **Generation of Candidate Responses**: This phase involves prompting the LLM to generate multiple candidate responses for a given input. Sensitivity analysis is performed on the responses to make sure determinism atleast some of it integrated in the response generation for avoiding computational costs
- **Aspect Extraction and Criteria Creation**: Relevant aspects of the responses are identified, such as coherence, relevance, and factual accuracy etc. Evaluation criteria are then established to ensure a standardized and consistent assessment process is carried out on them to see if any of the new criteria should be established for the response generation and if identified these are appended to the existing ones.
- **Criteria Matching with Aspects**: Each of the extracted aspect is matched with different criteria using a scoring system based on its relevancy or similarity.
- **Ranking Responses Based on Criteria**: Responses are ranked based on their performance against the established criteria and aspects present in them. A scoring mechanism, weighted by the importance of the criteria, is used to determine the relative quality of each response.

Note

Due to current computational resource limitations, this study is restricted to a select group of model providers. In future work, we plan to broaden the scope to encompass a more diverse set of model providers, including those tailored for specific applications and fine-tuned models. This paper introduces an abstract evaluation framework, and we welcome further contributions from the academic community to refine and expand this methodology.

3.1 Mathematical overview

Definition 1 (Query Space)**.** Let $\mathcal{Q}$ denote the query space, defined as the set of all possible queries that can be processed by an LLM. A query $x \in \mathcal{Q}$ is a sample query, represented as a structured input (e.g., a string, vector, or other encoding compatible with the LLM).

Definition 2 (Response Space)**.** Let $\mathcal{R}$ denote the response space, defined as the set of all possible responses generated by the LLM for queries in $\mathcal{Q}$. For a query $x \in \mathcal{Q}$, the LLM produces a set of candidate responses $Y \subseteq \mathcal{R}$.

Definition 3 (Perturbations)**.** For a query $x \in \mathcal{Q}$, let $\text{Pert}_i : \mathcal{Q} \to \mathcal{Q}$ for $i = 1, 2, \ldots, n$ denote a perturbation function that modifies x to produce a perturbed query $x_i = \text{Pert}_i(x)$. Perturbations are applied to enhance the determinism of responses and fix the parameters, all of which are listed below.

Table 1: LLM Parameters for Response Generation

Parameter	**Description**	**Value Type**	**Example Usage**
`num_ctx`	Context window size for next token generation. (Default: 2048)	int	`num_ctx 4096`
`repeat_last_n`	Look-back distance to prevent repetition. (Default: 64)	int	`repeat_last_n 64`
`repeat_penalty`	Penalty strength for repetitions. (Default: 1.1)	float	`repeat_penalty 1.1`
`temperature`	Randomness of token selection. (Default: 0.8)	float	`temperature 0.7`
`seed`	Random seed for consistent outputs. (Default: 0)	int	`seed 42`
`stop`	Stop sequences to halt generation.	string	`stop "AI assistant:"`
`num_predict`	Max tokens to predict. (Default: -1)	int	`num_predict 42`
`top_k`	Top k tokens for sampling. (Default: 40)	int	`top_k 40`
`top_p`	Cumulative probability threshold. (Default: 0.9)	float	`top_p 0.9`
`min_p`	Min probability filter. (Default: 0.0)	float	`min_p 0.05`

The model card allows for the perturbations of the above parameters the additional parameters such as presence penalty, frequency penalty, max length, min length, eta cutoff, epsilon penalty etc can be set by the user based on t6he model specs the code repo can be found in the appendix below

Definition 4 (Aspects)**.** Let $\mathcal{F}$ denote the aspect space, defined as the set of all possible aspects extractable from responses. For each response $y_i \in \mathcal{R}$, let $f_i \in \mathcal{F}$ denote an aspect extracted from y_i, where $f_i = f(y_i)$ and $f : \mathcal{R} \to \mathcal{F}$ is an aspect extraction function.

Definition 5 (Criteria Space)**.** Let $\mathcal{C}$ denote the criteria space, defined as the set of all possible criteria used to evaluate response similarity. A criterion $c \in \mathcal{C}$ represents a feature or property used to assess responses.

Definition 6 (Similarity Function)**.** Let Sim $: \mathcal{F} \times \mathcal{C} \to [0, 1]$ denote a similarity function computed by the LLM, where Sim(f_i, c) measures the similarity between an aspect $f_i \in \mathcal{F}$ and a criterion $c \in \mathcal{C}$.

Definition 7 (Response Ranking)**.** For a set of responses $Y = \{y_1, y_2, \ldots, y_n\}$, let Rank : $\mathcal{R}^n \to \mathbb{N}^n$ denote a ranking function that assigns an order to responses based on their similarity to criteria in $\mathcal{C}$. The ranking is determined by aggregating similarity scores.

Notation 1. The following notation is used:

- $\mathcal{Q}$: Query space.
- $\mathcal{R}$: Response space.
- $\mathcal{F}$: Aspect space.
- $\mathcal{C}$: Criteria space.
- $x \in \mathcal{Q}$: A sample query.
- $x_i = \text{Pert}_i(x)$: Perturbed query for $i = 1, 2, \ldots, n$.
- $Y = \{y_1, y_2, \ldots, y_n\} \subseteq \mathcal{R}$: Set of candidate responses.
- $f_i \in \mathcal{F}$: Aspect extracted from response y_i.
- $S = \{f_1, f_2, \ldots, f_n\} \subseteq \mathcal{F}$: Initial set of aspects.
- LLM: Large Language Model.
- Sim(f_i, c): Similarity score between aspect f_i and criterion c.
- $D = S_1 \cup S_2 \cup \cdots \cup S_n$: Union of aspect sets (if multiple sets are defined).

4 Formal Description

For a query $x \in \mathcal{Q}$, the LLM applies perturbations Pert$_i$ for $i = 1, 2, \ldots, n$, generating perturbed queries $x_i = \text{Pert}_i(x)$. Each x_i is processed by the LLM to produce a response $y_i \in \mathcal{R}$, forming the set of candidate responses:

$$Y = \{y_1, y_2, \ldots, y_n\}.$$

For each response $y_i \in Y$, an aspect extraction function $f : \mathcal{R} \to \mathcal{F}$ extracts an aspect $f_i = f(y_i)$. The initial set of aspects is:

$$S_1 = \{f_1, f_2, \ldots, f_n\}.$$

If additional aspect sets $S_2, \ldots, S_n$ are defined (e.g., from iterative processing or alternative extractions), let:

$$D = S_1 \cup S_2 \cup \cdots \cup S_n.$$

The LLM evaluates the similarity of each aspect $f_i \in S_1$ against a set of criteria $C \subseteq \mathcal{C}$. For each aspect f_i and criterion $c \in C$, the similarity score is:

$$s_i = \text{Sim}(f_i, c).$$

The set C is initially predefined but can be updated dynamically.

If a new class of criteria is identified during processing (e.g., via clustering, anomaly detection, or LLM analysis), a new criterion $c_{\text{new}} \in \mathcal{C}$ is appended to C:

$$C \leftarrow C \cup \{c_{\text{new}}\}.$$

Responses are judged using the ranking methods

Let responses $y_1, y_2, \ldots, y_n$ be ranked based on criteria $c_1, c_2, \ldots, c_n$. For each response y_i, the ranking function Rank orders responses by σ_i in descending order:

$$\text{Rank}(y_1, y_2, \ldots, y_n) = (r_1, r_2, \ldots, r_n),$$

where $r_i \in \mathbb{N}$ is the rank of y_i, and $\sigma_{r_i} \geq \sigma_{r_{i+1}}$.

Each of the criteria is classified as a positive or a negative criteria and based on that we evaluate the responses. To account for this we have introduced a row vector with $[-1, 1, -1 .... -1_n]$ where -1 is for negative criteria and the +1 for the positive criteria. because of the negative sign the ranking is reversed.

4.1 Analysis of ranking procedure

The following ranking models are applied:

Algorithm 1 TOPSIS Algorithm

1: **Input:** Decision matrix $X = [x_{ij}]_{m \times n}$, criteria weights $w = [w_j]_{1 \times n}$.
2: Construct the normalized decision matrix: $r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}$, for $i = 1, \ldots, m$, $j = 1, \ldots, n$.
3: Compute the weighted normalized matrix: $v_{ij} = w_j \cdot r_{ij}$.
4: Determine the positive ideal solution (PIS), negative ideal solution (NIS) $A^+, A^- = \{v_j^+ \mid v_j^+ = \max_i v_{ij}$ (benefit), $\min_i v_{ij}$ (cost)$\}, \{v_j^- \mid v_j^- = \min_i v_{ij}$ (benefit), $\max_i v_{ij}$ (cost)$\}$.
5: Calculate Euclidean distances: $S_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^+)^2}$, $S_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2}$.
6: Compute relative closeness: $C_i = \frac{S_i^-}{S_i^+ + S_i^-}$.
7: Rank alternatives by descending C_i.
8: **Output:** Ranked list of alternatives.

Algorithm 2 VIKOR Algorithm

Input: Decision matrix $X = [x_{ij}]_{m\times n}$, weights $w = [w_j]_{1\times n}$, strategy weight $v \in [0, 1]$.
Identify best and worst values: $f_j^+ = \max_i x_{ij}$ (benefit) or $\min_i x_{ij}$ (cost), $f_j^- = \min_i x_{ij}$ (benefit) or $\max_i x_{ij}$ (cost).
Compute utility measure: $S_i = \sum_{j=1}^{n} w_j \cdot \frac{f_j^+ - x_{ij}}{f_j^+ - f_j^-}$.
Compute regret measure: $R_i = \max_j \left(w_j \cdot \frac{f_j^+ - x_{ij}}{f_j^+ - f_j^-} \right)$.
Calculate VIKOR index: $Q_i = v \cdot \frac{S_i - \min_i S_i}{\max_i S_i - \min_i S_i} + (1 - v) \cdot \frac{R_i - \min_i R_i}{\max_i R_i - \min_i R_i}$.
Rank alternatives by ascending Q_i, S_i, R_i.
Check compromise conditions: If $Q(a_2) - Q(a_1) \geq \frac{1}{m-1}$ and a_1 is best in S_i or R_i, select a_1; else propose compromise set.
Output: Ranked list or compromise solution.

Algorithm 3 COPRAS Algorithm

Input: Decision matrix $X = [x_{ij}]_{m\times n}$, weights $w = [w_j]_{1\times n}$.
Normalize decision matrix: $r_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}}$.
Compute weighted normalized matrix: $d_{ij} = w_j \cdot r_{ij}$.
Calculate benefit score: $S_i^+ = \sum_{j\in\text{benefit}} d_{ij}$.
Calculate cost score: $S_i^- = \sum_{j\in\text{cost}} d_{ij}$.
Compute relative weight: $Q_i = S_i^+ + \frac{\sum_{i=1}^{m} S_i^-}{S_i^- \cdot \sum_{i=1}^{m} \frac{1}{S_i^-}}$.
Rank alternatives by descending Q_i.
Output: Ranked list of alternatives.

Algorithm 4 PROMETHEE I & II Algorithm

Input: Decision matrix $X = [x_{ij}]_{m\times n}$, weights $w = [w_j]_{1\times n}$, preference functions P_j.
Compute pairwise differences: $d_j(a, b) = x_{aj} - x_{bj}$ for each criterion j.
Apply preference functions: $P_j(a, b) = P_j(d_j(a, b))$.
Calculate preference index: $\pi(a, b) = \sum_{j=1}^{n} w_j \cdot P_j(a, b)$.
Compute positive outranking flow: $\phi^+(a) = \frac{1}{m-1} \sum_{b\neq a} \pi(a, b)$.
Compute negative outranking flow: $\phi^-(a) = \frac{1}{m-1} \sum_{b\neq a} \pi(b, a)$.
PROMETHEE I: Rank partially using $\phi^+(a)$ and $\phi^-(a)$ (outranking relations).
PROMETHEE II: Compute net flow $\phi(a) = \phi^+(a) - \phi^-(a)$ and rank by descending $\phi(a)$.
Output: Partial (I) or complete (II) ranking.

Algorithm 5 WASPAS Algorithm

1: **Input:** Decision matrix $X = [x_{ij}]_{m \times n}$, weights $w = [w_j]_{1 \times n}$, parameter $\lambda \in [0, 1]$.
2: Normalize decision matrix: $r_{ij} = \frac{x_{ij}}{\max_i x_{ij}}$ (benefit) or $r_{ij} = \frac{\min_i x_{ij}}{x_{ij}}$ (cost).
3: Compute WSM score: $Q_i^{(1)} = \sum_{j=1}^{n} w_j \cdot r_{ij}$.
4: Compute WPM score: $Q_i^{(2)} = \prod_{j=1}^{n} (r_{ij})^{w_j}$.
5: Calculate combined score: $Q_i = \lambda \cdot Q_i^{(1)} + (1 - \lambda) \cdot Q_i^{(2)}$.
6: Rank alternatives by descending Q_i.
7: **Output:** Ranked list of alternatives.

5 Approach

For evaluating the model responses we use a variety of libraries, and for the most efficient approach we are using ollama-based locally downloadable models for the experiments.