LLMatch Algorithm 2.0 Methodology

Ciara Adkins

May 2024

Abstract

This document outlines the methodology of the LLMatch Algorithm 2.0, detailing the mathematical framework and steps involved in the model ranking process.

For the code, visit: LLMatch POC on Google Colab.

Mathematical Framework of the LLMatch Model Ranking Algorithm

Notation and Definitions

Let:

- U be the user preference vector.
- M be the matrix whose rows M_i are the attribute vectors of each model.
- W be the weight vector applied to the attributes during the distance calculation.
- D be the vector of distances between the user preferences and each model.
- S be the final score vector for ranking the models.

Steps

1. User Preference Vector U The user preference vector U is constructed from specific user inputs, each mapped to corresponding model attributes. This vector represents the "ideal model vector" as envisioned by the user, where each component is a value reflecting the user's preferences regarding that attribute. The closer a model's attribute vector M_i is to U, the better the model aligns with the user's needs. The range for each component of U is either [0, 1] or [-1, 1], depending on the attribute, indicating preference intensity or direction.

Definition of U

$$U = [u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8]$$

where each component is mapped as follows:

- u_1: Type — Maps to whether the model is open-source or commercial. Values are:
  - -1 for strong preference towards open-source (when extra security is valued).
  - 0.5 for no strong preference or commercial.
- u_2: Parameters — Reflects preference regarding the model's computational complexity. Lower values are better when low energy usage is a priority.
- u_3: Freshness — Indicates the recency of model training. A value of -1 corresponds to a preference for the most recent models.
- u_4: Context Window — Represents the maximum combined input and output token count the model can handle.

- u_5: HumanEval — Corresponds to coding ability benchmarks, with higher values indicating a preference for models that are better at coding tasks.
- u_6: Inference Price — Relates to the cost-effectiveness of the model. Lower values are better for budget-conscious selections.
- u_7: Throughput — Tied to the processing speed of the model in tokens per second. Higher values indicate a preference for faster models.
- u_8: Latency — Time to the first token from the API; lower values are preferred for faster real-time interactions.

Representation of Preferences The values in U represent how closely a model must match specific attributes to be considered ideal:

- Values closer to 1 (or -1 where applicable) indicate stronger preferences for that attribute.
- A value of 0.5 (or 0 in some normalized scales) typically indicates neutrality or the average acceptable condition.

The goal is to find the model vector M_i that minimizes the distance D_i to U, which is calculated using the weighted Euclidean distance formula. This process effectively ranks models based on how well they align with the user-defined "ideal model vector".

2. Model Attribute Matrix M Each model in the dataset contributes a row vector to M:

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{k1} & m_{k2} & \dots & m_{kn} \end{bmatrix}$$

Where m_{ij} represents the j-th attribute of the i-th model, normalized and possibly filled for NaNs as discussed

3. Weight Vector W The weight vector W plays a critical role in the LLM-Match algorithm by adjusting the impact of each model attribute based on user preferences. If a user indicates that a particular attribute is irrelevant or less important, this preference is reflected by setting or reducing the corresponding weight in W. This mechanism ensures that the distance calculation prioritizes attributes that are more significant to the user, thereby refining the model selection process to better align with user needs.

Definition of W

$$W = [w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8]$$

where each component is defined as follows:

- w_1: Weight for Type (Security) — Doubled to emphasize the importance of security when selected by the user. A higher weight means that deviations from the preferred type (commercial or open-source) have a larger impact on the distance, making it a significant factor in model ranking.
- w_2: Weight for Parameters (Energy Usage) — Set to 1 or 0 based on whether energy usage is a concern, ensuring that models with undesirable parameter counts are penalized only if energy usage is important to the user.
- w_3: Weight for Freshness — Similar to other attributes, this weight is set based on user interest in model recency, effectively ignoring this attribute in the scoring if freshness is not a concern.
- w_4: Weight for Context Window — Always set to 1, as the context window size is generally always a consideration for model usability.
- w_5: Weight for HumanEval (Coding) — Quadrupled when coding ability is crucial, dramatically increasing the relevance of the HumanEval score in the overall distance calculation, which strongly favors models with superior coding performance.
- w_6: Weight for Inference Price (Budget) — Doubled to emphasize cost considerations more when budget is a key concern, thus making the price a decisive factor in the ranking.
- w_7 and w_8: Weights for Throughput and Latency (Speed) — Set based on the importance of speed, with higher weights making the performance metrics more pivotal in the evaluation process.

Operational Impact of W In the weighted Euclidean distance calculation:

$$D_i = \sqrt{\sum_{j=1}^{n} w_j (u_j - m_{ij})^2}$$

W directly modifies the contribution of each difference $(u_j - m_{ij})$ between the user preference vector U and each model vector M_i. This adjustment means that attributes aligned with user preferences will significantly influence model ranking, whereas those deemed irrelevant will have minimal or no impact.

This method ensures that the model ranking is not only based on how closely models match the user's specified conditions but also on how important each condition is to the user, making the ranking system adaptable and sensitive to individual user needs.

4. Distance Calculation The distance between the user vector and each model vector is calculated using a weighted Euclidean distance formula:

$$D_i = \sqrt{\sum_{j=1}^{n} w_j (u_j - m_{ij})^2}$$

This results in a distance vector D containing the distances for each model to the user preferences.

5. Scoring and Ranking

Normalization of MMLU and Distances To ensure that both model quality (represented by MMLU scores) and alignment with user preferences (represented by distances) are comparably influential, normalization steps are applied to both metrics before they are used in the scoring function.

- **MMLU Score Normalization (**$mmlu\_norm$**)**: The MMLU score, which assesses a model's performance on the HumanEval benchmark, is typically provided on a scale from 0 to 100. To normalize these scores to a range between 0 and 1, each score is divided by 100:

  $$mmlu\_norm_i = \frac{mmlu_i}{100}$$

  This normalization ensures that the MMLU score can be effectively integrated with other metrics that are also bounded between 0 and 1.

- **Distance Normalization (**$distance\_norm$**)**: The distances calculated using the weighted Euclidean formula are normalized across all models to ensure they range from 0 to 1. This is done by subtracting the minimum distance found across all models from each distance and then dividing by the range of the distances:

  $$distance\_norm_i = \frac{D_i - \min(D)}{\max(D) - \min(D)}$$

  After normalization, a transformation is applied to make lower distances more beneficial, as they indicate a closer match to user preferences:

  $$inverted\_distance\_norm_i = 1 - distance\_norm_i$$

  This transformation ensures that a smaller original distance (indicating a closer match to user preferences) results in a higher value in the final score calculation, thereby making it a more favorable factor.

Composite Scoring Function With both the MMLU score and the distances normalized, the final score for each model is computed using the following formula:

$$S_i = mmlu\_norm_i \times inverted\_distance\_norm_i$$

This scoring function integrates the normalized model quality and the inverted normalized distance to rank each model. Models with higher scores are those that not only perform well on the HumanEval benchmark but also align closely with the user's specified preferences, making them the most desirable recommendations.

Model Ranking Finally, models are ranked based on their scores S, with higher scores indicating better matches to user preferences. The models are sorted in descending order of their scores:

$$\text{Ranked Models} = \text{sort\_indices\_descending}(S)$$

This ordered list of models is then presented to the user, with the top models being those most aligned with both the user's preferences and high performance standards.