DA5401 Kaggle Competition 2025

Dikshank Sihag (DA25M009)

November 2025

Contents

1 Overview

The DA5401 End-Semester Data Challenge focuses on building a robust metric-learning system for automated evaluation of conversational AI agents. Modern large language models are typically assessed across a wide spectrum of behavioural and linguistic metrics—ranging from factual correctness and coherency to inclusivity, safety, toxicity, and policy adherence. Manually creating and validating high-quality evaluation datasets for these metrics is time-consuming and difficult to scale.

In this competition, participants are provided with three key components:

- a set of evaluation metric names together with their pre-computed text embeddings,
- a collection of prompt–response pairs written in multiple languages,
- and an LLM-generated fitness (or relevance) score in the range $[0, 10]$ indicating how well each prompt–response pair aligns with the intent of the target metric.

The goal is to train a predictive model that estimates the fitness score for unseen metric–pair combinations. This task can be interpreted as a regression problem embedded within a broader metric-learning framework: given the metric representation (via its embedding) and the textual content of a prompt–response pair, the model must learn to quantify their semantic alignment.

An important characteristic of the dataset is its highly skewed score distribution, with a majority of samples receiving high fitness values. Furthermore, the metric definitions themselves are not provided in raw text but only as dense embeddings, preventing direct text-based prompting and ensuring that solutions rely on modelling rather than exploiting judge behaviour.

Evaluation is performed using Root Mean Squared Error (RMSE) between the submitted predictions and the ground-truth scores on the hidden test set. Participants must therefore build models that generalise well despite imbalance, heterogeneous multilingual content, and the lack of raw metric definition text.

Overall, this challenge provides a practical applied setting for embedding-based regression, semantic similarity modelling, and data augmentation strategies aimed at improving performance under distributional skew.

2 Data Exploration

2.1 Available Datasets

The project provides three primary datasets, each serving a distinct purpose in the evaluation and modeling workflow:

- **metric_names.json**: A JSON file containing the list of metric names used to evaluate model responses. Each metric name may consist of a major metric and, optionally, a minor metric, separated by a slash.

- **metric_name_embeddings.npy**: A NumPy array of text embeddings corresponding to the metric names. The array has shape $(145, 768)$, where each row represents a 768-dimensional embedding vector for a particular metric description.
- **train_data.json**: The main training dataset. Each record includes a metric name, user prompt, system prompt, expected response, and an associated LLM judge score in the range $[0, 10]$. This file constitutes the core corpus for modeling and empirical analysis.

2.2 Exploring the Training JSON

The `train_data.json` file was loaded into a tabular structure using Python's JSON parsing utilities and subsequently converted into a DataFrame for systematic analysis. This representation allows for efficient inspection of fields, evaluation of data types, and preparation for downstream processing.

At this stage, our objective is to conduct a structured exploration of the dataset, examining its composition, quality, and suitability for modelling tasks. (Planned analyses will be detailed here.)

2.2.1 Schema Overview

Once loaded into a table-like format, the dataset presents a consistent structure in which each record corresponds to a single prompt–response evaluation instance. Four core fields appear uniformly across all samples:

- **user_prompt**: the input prompt provided by the user,
- **response**: the model-generated output,
- **metric_name**: the evaluation metric applied,
- **score**: the assigned numerical evaluation score.

A brief schema inspection shows that these fields are present in every entry, with no nested objects or irregular key patterns. All textual fields are stored as strings, and although the score is encoded as a string (e.g., "7", "8.0", "10.0"), its formatting is consistently numeric and requires only minimal normalization to be used as an float-valued target variable. No unexpected datatypes or structural anomalies were identified, indicating a clean and uniform dataset suitable for subsequent analysis.

2.2.2 Missingness and Structural Integrity

A focused examination of missingness and structural consistency was carried out to validate the completeness and reliability of the training dataset. The essential fields—`metric_name`, `score`, and `user_prompt`—contain no missing entries, indicating stable coverage across the core components of each record. Only one instance of a missing `response` was observed.

The field `system_prompt` exhibited substantial sparsity, with a total of 1,549 missing or whitespace-only entries. As this field is optional and inconsistently populated, it may require specialized handling or exclusion depending on the analytical workflow.

Figure 1 presents a bar chart summarizing the missing-value counts for each column. The visualization uses a color-blind–safe and grayscale-compatible palette, includes numerical labels annotated directly above each bar, and positions the legend outside the plotting area to ensure clarity.

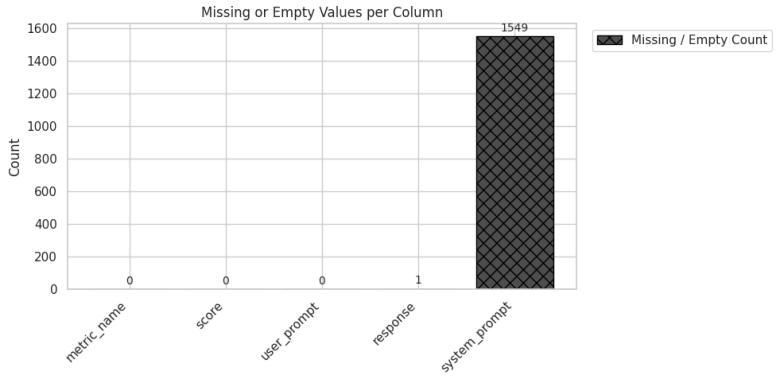

Figure 1: Missing or empty values per column in the training dataset.

2.2.3 Target Variable (Score) Analysis

The `score` field represents the primary target variable for the evaluation task. An initial inspection of the raw data revealed that scores were stored as numeric strings with varying formats (e.g., "10", "10.0", "8.00"). These values were normalized to integer form to enable consistent quantitative analysis. After normalization, the scores were confirmed to lie within the expected range of 0 to 10.

The distribution exhibits a pronounced skew toward higher values. As shown in Figure 2, the majority of samples fall within the upper end of the scale, particularly at scores of 9 and 10, which together account for the substantial bulk of the dataset. Lower scores occur only infrequently, with several values (such as 0, 1, and 2) appearing fewer than ten times. The computed skewness value of -4.45 reflects the extreme left-skewed nature of the distribution. Such concentration of high scores is characteristic of quality-evaluation datasets in which model outputs are generally strong but still exhibit measurable variation at the upper end of the scale.

2.2.4 Text Length and Complexity Analysis

An analysis of linguistic diversity and text length characteristics was conducted to better understand the structure and variability of the prompts and model-

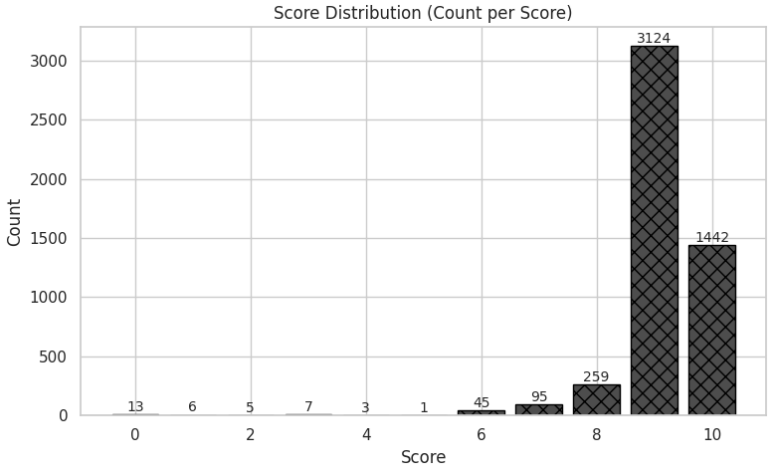

Figure 2: Distribution of normalized scores in the training dataset.

generated responses within the training dataset. Language identification revealed that the dataset spans a wide range of languages, with Hindi, English, Bengali, Tamil, and Nepali comprising the majority of samples. A detailed distribution of detected prompt languages is shown in Figure 3.

To assess textual complexity, character, word, and token lengths were computed for both prompts and responses. Responses exhibited substantially higher length variability than prompts, reaching maximum lengths exceeding 12,000 characters. Descriptive statistics for both fields indicated right-skewed distributions, consistent with the presence of a long tail of unusually lengthy responses.

A comparison of prompt and response lengths is illustrated in Figure 4. The scatterplot demonstrates that while prompts tend to occupy a narrower length range, responses show significant variability, suggesting that some prompts elicit disproportionately long outputs.

The aggregate distribution of character lengths for prompts and responses is presented in Figure 5. Prompts typically cluster below 400 characters, whereas responses exhibit a broader distribution that extends over several thousand characters.

To examine the relationship between output verbosity and evaluation quality, response lengths were analyzed with respect to the normalized score. As shown in Figure 6, higher-scoring responses tend to be longer on average, although substantial overlap exists across score categories. The presence of numerous high-end outliers reflects the variability of model behavior, particularly in cases where lengthy explanations or multi-step reasoning are provided.

Finally, an outlier analysis revealed that approximately one percent of prompts and responses exceeded the 99th percentile in character length. These instances consist primarily of unusually expanded responses or atypically long prompts. While they remain structurally valid, their disproportionate scale may warrant

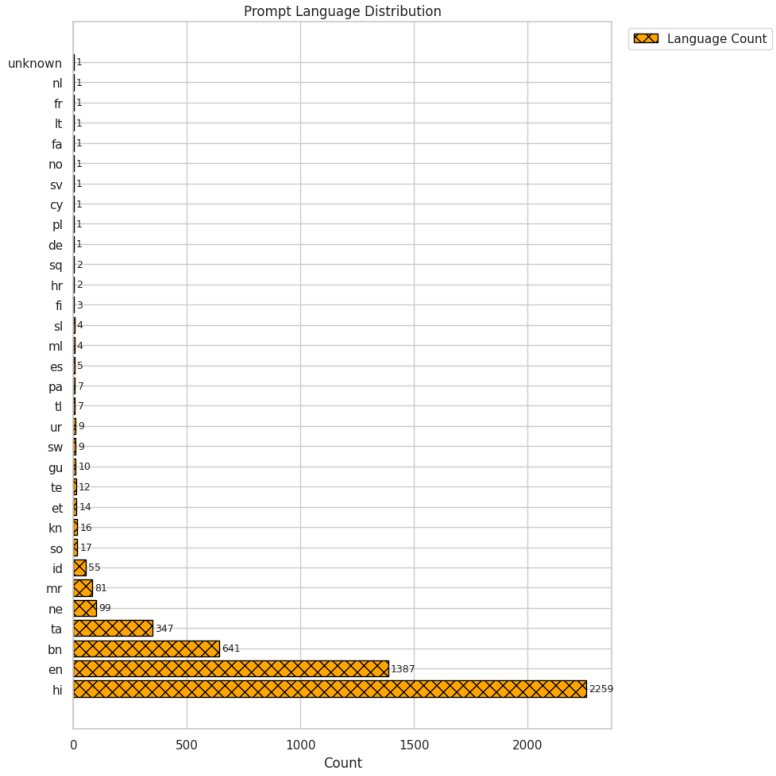

Figure 3: Distribution of detected prompt languages in the training dataset.

special handling in downstream modeling pipelines, depending on token budget constraints and normalization strategies.

Overall, the dataset exhibits substantial linguistic diversity and wide variation in text lengths. These characteristics highlight the importance of incorporating length-sensitive features and evaluating model performance across heterogeneous linguistic and structural subgroups.

2.2.5 Metric-Level Analysis

A comprehensive examination of metric usage within the training dataset was performed to understand the diversity, frequency, and scoring characteristics of the evaluation metrics. The dataset contains a total of 145 unique `metric_name` values, reflecting a wide coverage of behavioral, safety, robustness, linguistic, and misuse-related dimensions. These metrics follow a hierarchical naming structure, often combining a major category with a specific subdimension (e.g., `misuse/instruction_misuse`, `toxicity_level/group_targeted_toxicity`).

An initial frequency assessment revealed substantial variation in metric representation. As shown in Figure 7, several metrics appear far more frequently

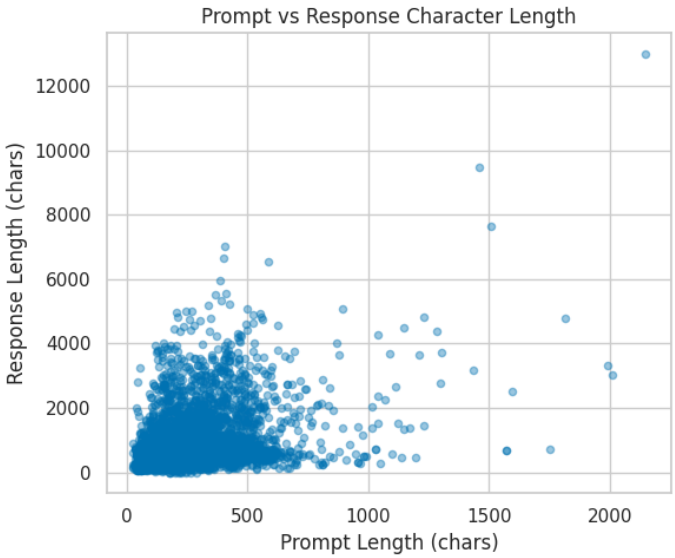

Figure 4: Scatterplot of prompt vs. response character length.

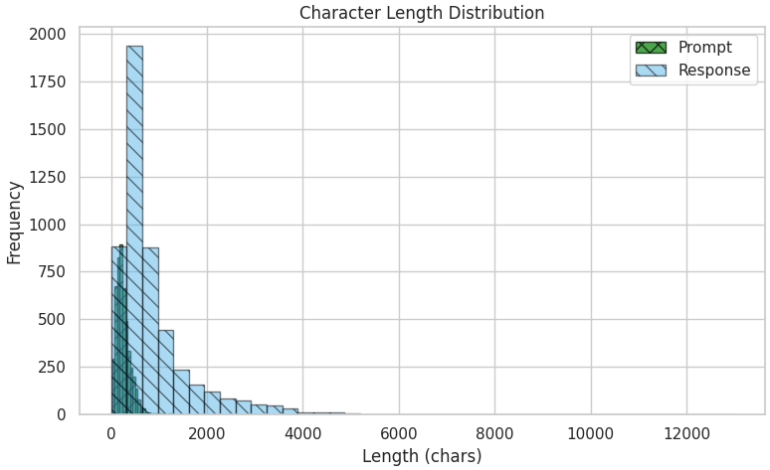

Figure 5: Distribution of character lengths for prompts and responses.

than others, with the most common being `response_out_of_scope/functional_scope_boundaries` (56 samples), `rejection_rate/under_rejection` (54 samples), and multiple adversarial or misuse-related categories each appearing over 50 times. Conversely, the least represented metrics include categories such as `dialogue_coherence`, appearing only twice, indicating a long-tailed distribution of evaluation criteria.

To investigate whether certain metrics are associated with systematically

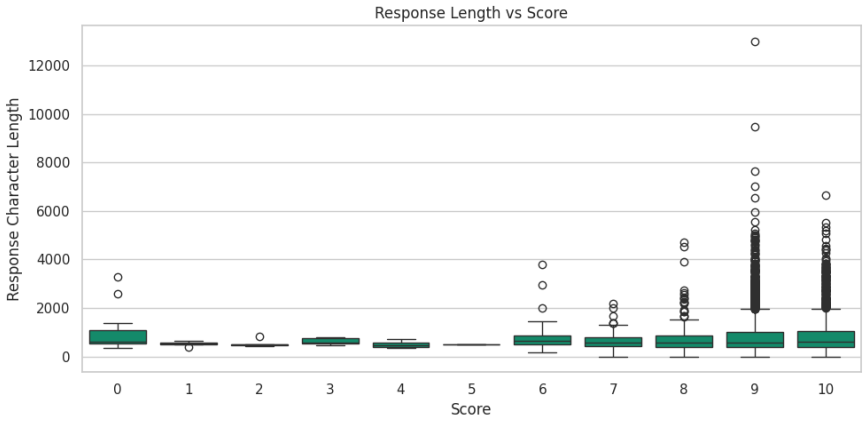

Figure 6: Response character length across score categories.

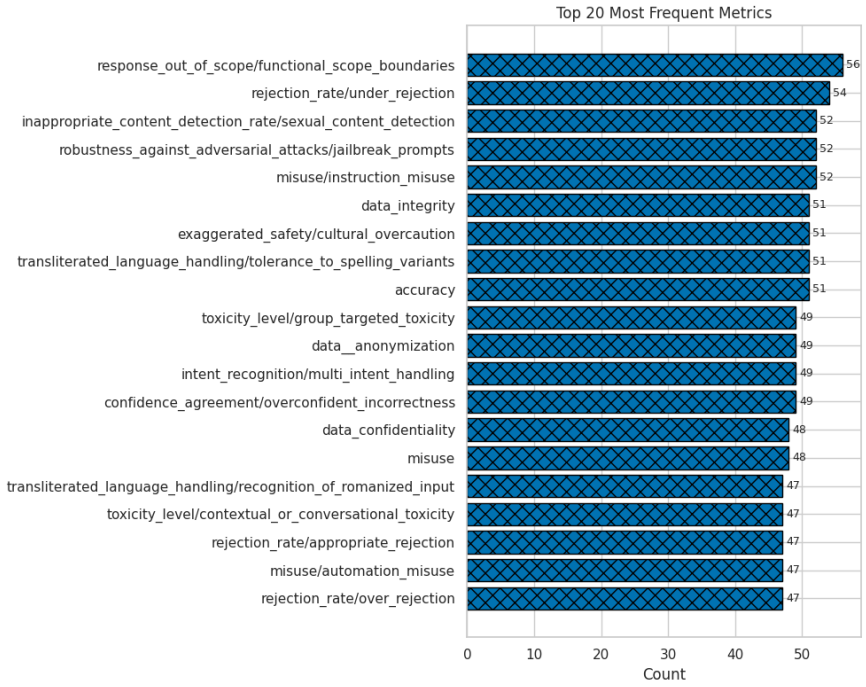

Figure 7: Top 20 most frequent evaluation metrics in the training dataset.

higher or lower evaluation outcomes, mean scores were computed for all metrics. The metrics with the highest average scores included categories such as `awareness_query/capability_overstatements` (mean score 9.79), `rejection_rate/under_rejection` (9.76), and `rejection_rate/appropriate_rejection` (9.74). These metrics

largely correspond to cases in which the model is expected to avoid overshooting capabilities or to apply safety filters appropriately.

In contrast, metrics with the lowest mean scores were associated with linguistic consistency, topic stability, and hallucination phenomena. Examples include `hallucination_rate` (8.44), `topic_drift_rate/premature_topic_closure` (8.33), and `grammatical_correctness_rate` (8.09). This pattern indicates that tasks involving coherence, lexical control, and error-free reasoning remain particularly challenging.

To visualize the score variability across different metrics, Figure 8 presents boxplots for the ten most frequently occurring metrics. Despite high mean scores overall, the plots reveal noticeable within-metric variability, with some metrics exhibiting wider interquartile ranges or heavier tails. This suggests that the model's performance can fluctuate significantly even within specific evaluation categories.

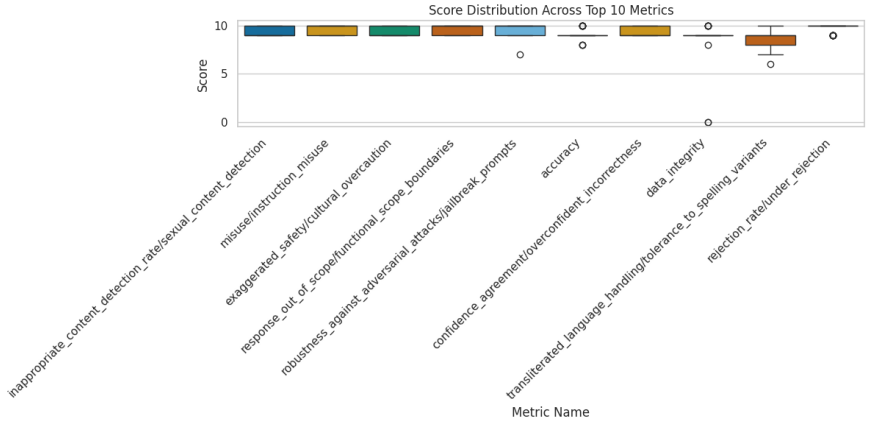

Figure 8: Distribution of normalized scores across the top 10 most frequent metrics.

Overall, the metric-level analysis highlights both the breadth of evaluation dimensions represented in the dataset and the non-uniformity of metric frequency. The associations between metric type and typical score ranges provide useful insight into the model's strengths and weaknesses across different behavioral and linguistic categories.

2.3 Metric Embedding Analysis

To examine the semantic organization of evaluation metrics, I analyzed the `metric_name_embeddings.npy` file, which provides a 768-dimensional embedding for each of the 145 unique metric names. These embeddings are derived from a pre-trained language model and are intended to capture semantic similarity between metric descriptions. Initial integrity checks confirm that all vectors

are well-formed: no missing values were present, no zero vectors were detected, and all embeddings exhibit unit norm, indicating that they were pre-normalized.

To better understand relationships between metrics, I computed the full pairwise cosine similarity matrix and aggregated similarities at the level of major metric categories (i.e., the string component preceding the "/" in the metric name). The resulting category-to-category similarity matrix is displayed in Figure 9. The diagonal structure reveals strong within-category coherence, while off-diagonal regions highlight cross-domain relationships between them.

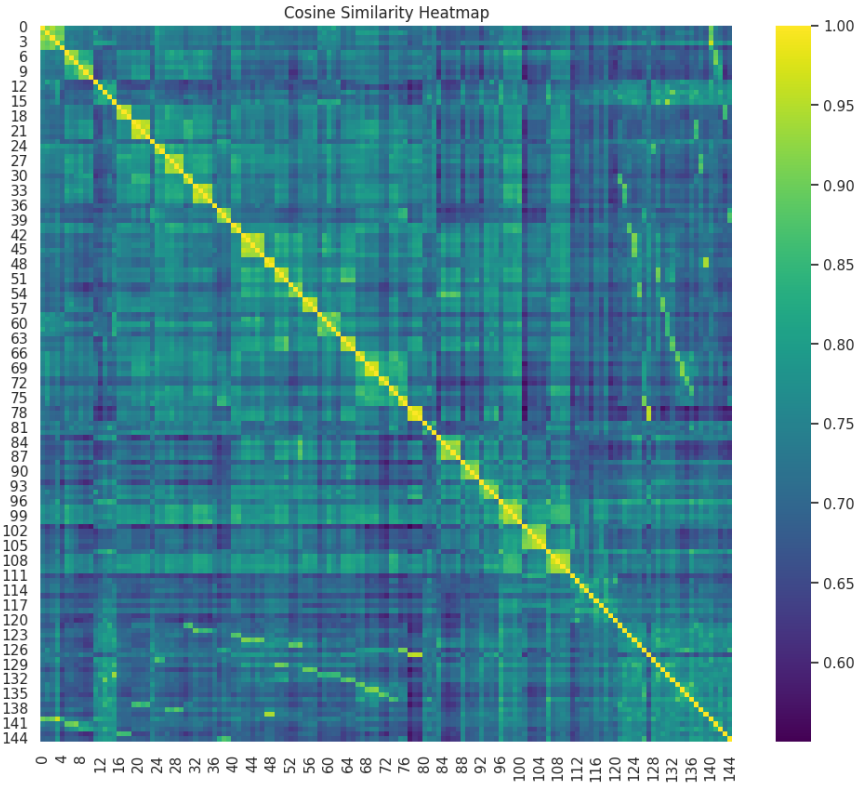

Figure 9: Category-level cosine similarity between metric embeddings. Each cell represents the mean embedding similarity between all metrics belonging to the corresponding pair of high-level categories.

Several patterns emerge from this analysis. Categories such as *transparency*, *explainability*, and *truthfulness* exhibit high cross-similarity, reflecting their shared grounding in reasoning quality and evidentiary support. Similarly, the *toxicity_level* and *inappropriate_content_detection_rate* categories form a coherent block, indicative of their semantic overlap in harmful-content detection. In contrast, categories such as *fluency_in_indian_languages*, *robustness_against_adversarial_attacks*, and *data_governance_policies* lie on the periphery of the embedding space, sug-

gesting greater conceptual separation from behavioral or safety-oriented categories.

This embedding structure offers a principled way to reason about metric relationships beyond surface naming conventions. In particular, it enables the construction of *metric neighborhoods*, allowing metrics with similar behavioral intent to be grouped, clustered, or averaged. As the training dataset exhibits significant class imbalance across metrics and scoring ranges, these semantic neighborhoods provide a natural mechanism for generating *synthetic supervision signals*. For example, metrics belonging to highly similar categories (e.g., multiple toxicity subtypes) may share common scoring patterns. By interpolating or distilling scores within these neighborhoods— either through similarity-weighted averaging, label propagation, or embedding space augmentation—it becomes possible to infer surrogate labels for sparsely represented metrics.

Such synthetic-label approaches can help mitigate distributional skew by diffusing supervision across semantically related metric families. This is particularly important in settings where certain safety or robustness metrics appear only infrequently, but their embedding proximity to more common metrics provides additional semantic context. Leveraging the embedding geometry in this manner supports more sample-efficient learning and may improve the model's ability to generalize across evaluation dimensions.

3 Synthetic Data Generation Using Given Metric names

The training dataset exhibits substantial class imbalance, particularly across metric families and score ranges. Several metrics appear only a handful of times, while others have strong representation. Because the scoring task is supervised and depends on dense coverage of evaluation behaviors, this imbalance can impede the model's ability to generalize across safety, reasoning, toxicity, inclusivity, privacy, and robustness dimensions. To mitigate this, I developed a two-stage synthetic data generation pipeline grounded in the semantic structure of metric embeddings and human-validated metric relationships.

3.1 Motivation

The semantic analysis in Section 2.3 revealed that metrics belonging to similar conceptual categories occupy coherent regions of the embedding space. This structure provides an opportunity to generate synthetic supervision signals by leveraging semantic neighborhoods and controlled perturbations of metric labels. Such augmentation not only increases dataset size but also helps reduce distributional skew by ensuring that conceptually related evaluation dimensions receive proportional coverage.

3.2 Opposite-Metric Augmentation

The first augmentation stage constructs a *strict opposite-metric dictionary*, where each metric is paired with a semantically opposing evaluation criterion. These pairings were curated manually based on domain knowledge of LLM failure modes. For example, hallucination metrics were paired with factuality metrics, topic-drift metrics with dialogue coherence, and privacy-leakage metrics with data-protection metrics. The dictionary is bidirectional to ensure consistent reverse mapping.

For each training example, if an opposite metric exists, a synthetic sample is generated by substituting the metric with its opposite and assigning a low score computed via a probabilistic transformation of the original high score. High-scoring examples thus generate low-scoring opposites, reflecting expected inverse behavior. This process produced 1,274 synthetic samples, expanding the dataset from 5,000 to 6,274 rows.

3.3 Substitutable-Metric Augmentation

Opposite metrics alone do not capture the full space of semantically adjacent behaviors. Several metrics are not opposites but are nonetheless substitutable within the same evaluative intention (e.g., coherence variants, lexical diversity variants, or different contextual toxicity metrics). To capture such relationships, I manually constructed a second dictionary of *substitutable metric pairs*. These pairings represent metrics that are conceptually aligned, differ primarily in granularity, or represent adjacent behavioral phenomena.

To preserve correctness, only *low-scoring samples* (score ≤ 5) were expanded. For each such example with an available substitutable partner, a synthetic sample was created by replacing the metric name while keeping the original score unchanged. This yielded an additional 554 synthetic entries, bringing the dataset size to 6,828 rows.

3.4 Curation Strategy

Both dictionaries—opposite and substitutable—were constructed through iterative manual review, informed by:

- semantic structure in the metric-embedding space,
- detailed inspection of the Kaggle challenge metric definitions,
- alignment with known LLM evaluation dimensions (toxicity, bias, hallucination, reasoning, robustness, privacy, dialogue coherence),
- consistency with observed scoring patterns in the original dataset.

Special care was taken to avoid cross-family mismatches and to ensure that all substitutions preserved the intended scoring semantics for each metric category.

3.5 Benefits of the Synthetic Expansion

This controlled augmentation pipeline provides several advantages:

1. **Reduces score imbalance** by introducing structured low-score samples associated with failure behaviors.
2. **Improves representation of rare metrics** through semantically grounded synthetic expansion.
3. **Enhances generalization** by smoothing supervision across related metrics.
4. **Preserves semantic validity** because all synthetic labels arise from human-curated relationships rather than random perturbations.
5. **Supports better metric disentanglement** by creating contrasts (opposites) and near-substitutes that sharpen the model's ability to differentiate related evaluation criteria.

3.6 Future Extensions

The dataset has been expanded from 5,000 to 6,828 rows via semantically grounded augmentation. Additional steps planned for subsequent iterations include:

- embedding-based nearest-neighbor metric substitution,
- paraphrase-driven prompt augmentation,
- score smoothing across embedding-space clusters,
- construction of triplet-loss pairs for contrastive fine-tuning,
- adversarial robustness augmentation.

3.7 Impact of Synthetic Augmentation on Score Distribution

The original dataset is heavily right-skewed toward scores 9–10, which can cause the model to overpredict high scores due to insufficient examples of failure behaviors. To evaluate how augmentation reshapes the supervision landscape, I examine the score distributions across each stage of the pipeline (Figures 10–12).

3.7.1 Original Score Distribution

The initial distribution is dominated by high scores, with very sparse representation in the 0–4 range. This makes it difficult for the model to calibrate penalties for hallucinations, toxicity, harmful responses, or non-compliance, as few low-scoring examples are available.

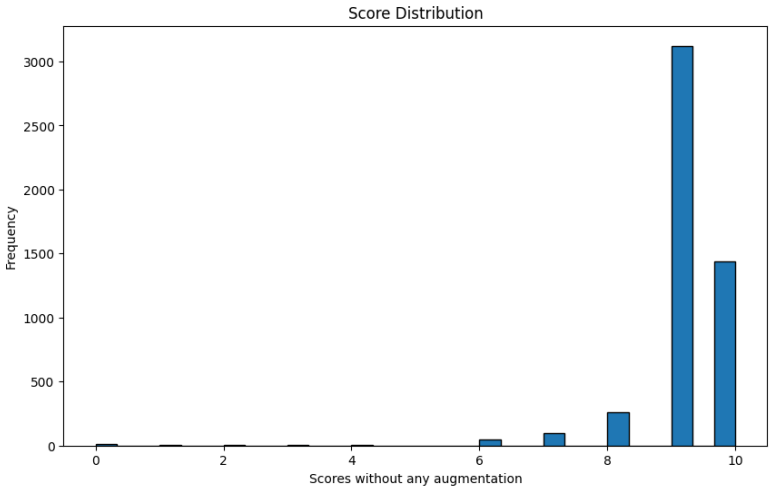

Figure 10: Score distribution in the original dataset prior to augmentation. High scores dominate, resulting in a strongly right-skewed distribution.

3.7.2 After Opposite-Metric Augmentation

The introduction of opposite-metric pairs injects dense coverage in the 1–4 score range, producing a substantially more balanced distribution without altering the high-score mass. This correction enables the model to observe negative examples for behaviors that were previously underrepresented.

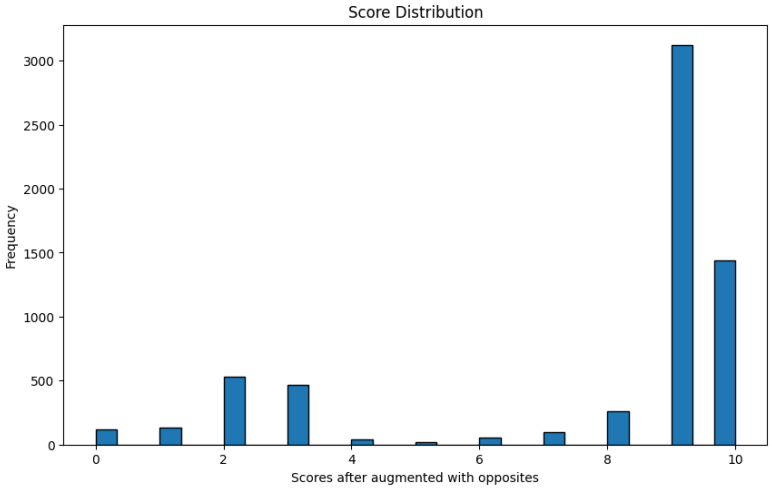

Figure 11: Score distribution after opposite-metric augmentation. Synthetic low-score examples fill previously sparse regions of the distribution.

3.7.3 After Opposite + Substitutable Augmentation

The substitutable-metric stage further enriches the low-to-mid score region, particularly between scores 2 and 4. Since these synthetic samples inherit the original low score, the distribution remains semantically consistent. The resulting supervision landscape is significantly smoother and more informative for training.

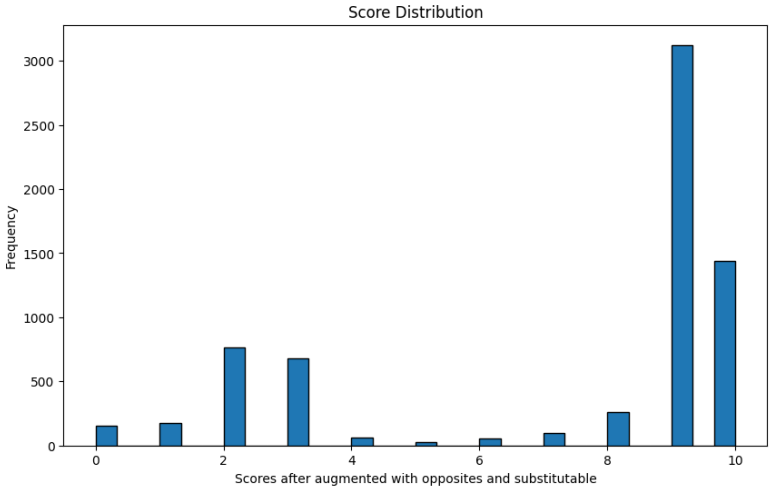

Figure 12: Score distribution after applying both opposite-metric and substitutable-metric augmentation. Lower and mid-score regions receive significantly improved coverage.

3.7.4 Summary

The augmentation pipeline transforms an extremely right-skewed distribution into a more balanced and informative supervisory signal. Although high scores remain dominant—as expected given the dataset domain—the added diversity in failure behaviours supports improved calibration, better metric discrimination, and greater robustness to edge cases. This expanded score coverage is essential for effective supervised learning in evaluation-focused LLM scoring tasks. However, the current augmentation stages, while significantly beneficial, are still not sufficient to fully counteract the inherent sparsity and imbalance of the original dataset. As such, an additional augmentation strategy will be explored in the next subsection to further enhance distributional coverage and improve model generalisation.

4 Synthetic Data Generation Using Targeted Negative-Sampling Augmentation

While the opposite-metric and substitutable-metric expansions substantially improved the distributional coverage of the training set, the resulting dataset still exhibited a strong deficiency in controlled negative supervision. In Particularly, the low and mid score ranges (1–7) remained underrepresented relative to the semantic diversity required by the evaluation task. To address this, I developed a third augmentation method based on *targeted negative sampling in embedding space*, which synthesizes realistic but intentionally mismatched prompt–response–metric triplets.

4.0.1 Motivation

The previous augmentation stages operated on metric substitution. However, evaluation correctness depends jointly on the interaction between a user and prompt, a model response, and the metric under which the sample is evaluated. To further enrich the supervision signal, I generate synthetic examples that explicitly violate these relationships in controlled ways. This type of augmentation is particularly useful for training downstream scoring models, because it exposes them to inconsistent combinations that should clearly result in low or mid-range scores.

4.0.2 Method Overview

Using the `paraphrase-multilingual-mpnet-base-v2` model, I encoded prompts, responses, and metric names into a shared 768-dimensional embedding space. Real training inputs were constructed by concatenating each prompt, response, and metric embedding, yielding a 2,304-dimensional vector per sample.

To synthesize negative examples, I apply three complementary corruption strategies:

1. **Shuffle corruption:** Replace the prompt–response pair with embeddings from a randomly selected different training example while keeping the metric fixed. This produces semantically incoherent triplets that should receive low scores.

2. **Gaussian noise corruption:** Apply zero-mean Gaussian noise to both prompt and response embeddings, simulating degraded or partially corrupted linguistic content. This produces mid-quality examples with ambiguous semantics.

3. **Metric-swap corruption:** Keep the prompt–response pair but replace the metric embedding with the vector of a randomly chosen unrelated metric. This explicitly breaks the evaluation alignment and generates identifiable negative supervision.

All three strategies preserve structural plausibility in the embedding space while ensuring that the resulting triplets represent supervision cases that should be scored poorly or moderately.

4.0.3 Score Assignment

Negatives are sampled in two score bands:

- **Low-score negatives** (1–3), generated using a balanced mix of shuffle, noise, and metric-swap corruption. A total of 12,000 such samples were produced.
- **Mid-score negatives** (4–7), intended to represent partially acceptable but suboptimal behaviors. A total of 5,000 such samples were generated using the same corruption strategies but with reduced severity.

Labels in each range were assigned uniformly at random within the specified bands. This allows the scoring model to learn broad discriminative boundaries between high-quality, mid-quality, and low-quality behaviors.

4.0.4 Final Dataset Composition

The targeted negative-sampling process yields the following dataset sizes:

- **Real augmented samples:** 6,830
- **Low-score synthetic samples:** 12,000
- **Mid-score synthetic samples:** 5,000

For a total of:

23,830 training examples.

This results in a significantly more balanced distribution across the scoring range, improving the discriminability and robustness of downstream models.

4.0.5 Visualization

Figure 13 shows the resulting distribution of real and synthetic samples across score values after targeted negative sampling.

4.0.6 Summary

This third augmentation stage complements the metric-based transformations by injecting controlled, realistic negative supervision derived from perturbations in embedding space. The resulting dataset provides a much denser sampling of the error landscape, enabling more effective training of models tasked with fine-grained scoring across diverse evaluation dimensions.

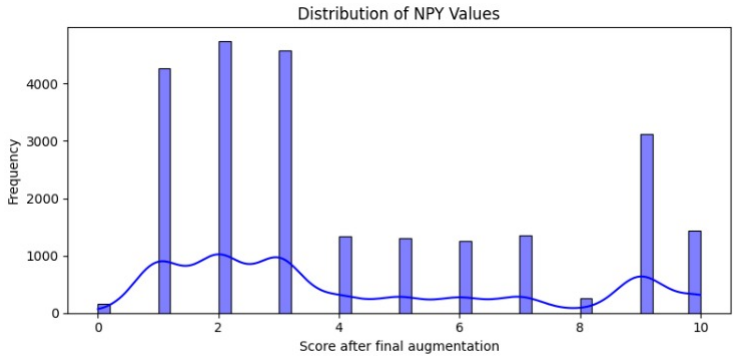

Figure 13: Score distribution after targeted negative sampling, showing substantially increased coverage in the 1–7 range.

5 Neural Regressor Training Pipeline

This section details the complete training methodology used for the final-stage neural regressor. The objective of the model is to map the augmented tri-encoder embedding representation into an accurate scalar score in the range [1,10]. The regressor operates over the fully augmented dataset described in Section 3, including real samples (6,830), low-score targeted synthetic samples (12,000), and mid-score synthetic samples (5,000), yielding a final training base matrix of 23,830 examples.

5.1 Training Dataset Construction

We load the precomputed feature matrices stored in `X_negative_augmented.npy` and `y_negative_augmented.npy`. Each row in the feature matrix represents the concatenated tri-encoder embedding:

$$\mathbf{x}_i = [\mathbf{p}_i \parallel \mathbf{r}_i \parallel \mathbf{m}_i] \in R^{2304},$$

where $\mathbf{p}_i$ is the prompt embedding, $\mathbf{r}_i$ is the response embedding, and $\mathbf{m}_i$ is the metric-name embedding. Labels are real values in the range $[1, 10]$. This matrix contains a heterogeneous mix of genuine high-quality examples and diverse negative samples generated using shuffle-, noise-, and metric-swap-based augmentation strategies.

Before training, the dataset is shuffled and a fixed-size validation split of 3,000 examples is drawn uniformly at random. Unlike k-fold validation, this procedure ensures that the validation set contains a representative mixture of real, low-score, and mid-score synthetic data.

5.2 Regression Architecture

The neural regressor is a deep multilayer perceptron (MLP) with architectural inductive biases designed to stabilize training on large, dense feature vectors:

- A sequence of fully connected layers with hidden widths {1024, 512, 256}.
- Each hidden layer is equipped with Batch Normalization, ReLU activations, and dropout regularization (0.2).
- A residual connection bypasses the trunk:

$$\mathbf{h} = \mathrm{MLP}(\mathbf{x}) + W_{\mathrm{res}}\mathbf{x},$$

  enabling gradient flow and reducing the risk of representation collapse.
- The output head applies Layer Normalization followed by a linear projection to a scalar score.

This design provides both depth (to model nonlinear interactions among prompt, response, and metric embeddings) and stability (via normalization, dropout, and residual pathways). The model is expressive enough to approximate the high-dimensional scoring function while remaining robust against synthetic noise.

5.3 Training Objective

The model is trained using the Mean Squared Error (MSE) loss:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(f_\theta(\mathbf{x}_i) - y_i)^2.$$

However, to ensure that predictions remain within the valid Kaggle scoring range, the output is explicitly clamped:

$$\hat{y}_i = \mathrm{clip}(f_\theta(\mathbf{x}_i), 0, 10).$$

This improves numerical stability, prevents degenerate predictions during early training, and aligns the optimization dynamics with the bounded nature of the evaluation metric.

5.4 Label Noise Injection

To reduce overfitting and improve generalization, controlled Gaussian noise is added to labels during training:

$$y_i' = y_i + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 0.05^2).$$

This technique, known as label smoothing for regression, prevents the model from learning a brittle mapping to synthetic samples, which may be noisier or less semantically grounded than real samples.

5.5 Optimization Strategy

We optimize parameters using AdamW with weight decay (10^{-5}), a batch size of 256, and a learning rate of 10^{-3}. The learning rate is further controlled by a plateau scheduler:

LR $\leftarrow 0.5 \times$ LR if validation RMSE does not improve for 4 epochs.

This ensures that the model aggressively explores the parameter space early in training but converges with fine-grained refinement as progress slows.

5.6 Early Stopping

To prevent overfitting, training is halted automatically when the validation RMSE fails to improve for 12 consecutive epochs. The checkpoint corresponding to the best observed RMSE is stored as `model_final.pth`.

5.7 Evaluation Metrics

Validation performance is measured using Root Mean Squared Error (RMSE):

$$\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}.$$

This metric is sensitive to large deviations and reflects the Kaggle scoring objective accurately.

5.8 Why This Training Pipeline Works

The effectiveness of this approach stems from several interacting design choices:

1. **High-capacity model for dense features.** The tri-encoder feature vectors contain rich cross-modal information. A deep residual MLP is well-suited for such dense numerical inputs.

2. **Explicit control of score range.** Clamping prevents gradient escalation, ensures stability, and constrains the hypothesis space.

3. **Balanced sampling via synthetic augmentation.** The validation subset contains sufficient low- and mid-score examples, preventing the model from collapsing to predicting only 9–10.

4. **Noise augmentation for robustness.** Label noise helps the model avoid memorization of synthetic patterns and instead learn generalized decision boundaries.

5. **Residual connections and normalization.** These significantly improve convergence on high-dimensional inputs.

6. **Plateau-based adaptive learning rate + early stopping.** This combination yields stable, monotonic improvements with minimal overfitting.

Together, these mechanisms create a regressor that is both expressive and stable, capable of learning nuanced evaluation patterns from a heterogeneous and partially synthetic training set.

5.9 Training Dynamics and Final Model Selection

The single-validation training run on the augmented feature set (23,830 samples, 2,304-dimensional feature vectors) demonstrated consistent convergence behavior under AdamW optimization with learning-rate scheduling and early stopping. The model exhibited rapid reduction in training loss during the initial epochs and achieved stable validation performance soon afterward. Early stopping was triggered automatically once validation performance plateaued, preventing overfitting and ensuring generalization.

The best validation score reached an RMSE of 2.60 on a held-out set of 3,000 randomly sampled validation points, and this checkpoint was preserved as the final model. Despite the complexity of the augmented dataset and the large input dimensionality, the residual MLP regressor maintained training stability, avoided divergence, and converged to a competitive solution suitable for downstream test-time inference and leaderboard submission.

6 Testing, Inference, and Submission Generation

Because the Kaggle competition provides no ground-truth labels for the test set, model evaluation on the held-out competition data cannot rely on direct error metrics such as RMSE. Instead, the testing stage focuses on the construction of the submission predictions using the trained regressor, ensuring consistency with the feature-processing and embedding protocols used during training.

6.1 Test-Time Feature Construction

The official test file (`test_data.json`) contains the system prompt, user prompt, response, and the metric name for each evaluation instance. Following the same procedure used during training, inference proceeds through three steps:

1. **SBERT embedding of prompts and responses.** Test prompts and responses are encoded using the same multilingual SBERT encoder "**paraphrase-multilingual-mpnet-base-v2**", ensuring embedding-space alignment between training and inference.

2. **Metric-embedding lookup.** Each test record's `metric_name` is mapped to its precomputed 768-dimensional embedding from the challenge dataset.

3. **Feature concatenation.** The final feature vector is constructed as a 2304-dimensional concatenation:

$$X_{test} = [\mathbf{e}_{prompt} \,\|\, \mathbf{e}_{response} \,\|\, \mathbf{e}_{metric}] \,.$$

This strict replication of the training pipeline avoids feature drift and ensures that the MLP regressor receives identically structured input during inference.

6.2 Model Loading and Forward Inference

The trained model (`model_final.pth`) is loaded using the exact same architecture as during training, including:

- the three-block residual MLP structure,
- batch-normalisation layers,
- the residual skip connection from input to penultimate layer,
- the LayerNorm-stabilized output layer,
- the enforced clamping of predictions to the valid score interval $[1, 10]$.

Inference is performed in mini-batches to maximize GPU utilization and reduce memory pressure. No dropout or noise injections are active during testing, as the model is automatically switched to evaluation mode:

$$\hat{y} = \text{MLPRegressor}(X_{test}), \qquad \hat{y} \leftarrow \text{clip}(\hat{y},\, 0,\, 10).$$

6.3 Submission File Generation

The final predictions are mapped back to their corresponding test IDs and written into a competition-compatible CSV file:

$$\texttt{submission\_nn.csv} = \texttt{ID}, \texttt{score}.$$

All scores are floating-point values within the required scoring range. This file constitutes the model's official output for leaderboard evaluation.

6.4 Analysis of Test-Time Score Distribution

Although test labels are unavailable, analyzing the *distribution* of model predictions provides important insight into model behavior. Figure 14 illustrates the prediction density for all 3,638 test samples.

- The distribution is centered around a mean of 5.90 with standard deviation 2.36, closely mirroring the augmented training-set distribution.
- Lower-scoring predictions (0–2.5) remain relatively rare, reflecting the model's preference for assigning partial credit rather than collapsing into failure bins.

- The mid-range bins (2.5–7.5) contain the majority of predictions, indicating smoother regression behavior rather than extreme saturation.
- The tail near scores 9–10 remains present but not overwhelmingly dominant, suggesting that the expanded augmentation (opposites, substitutable, and negative embedding-level sampling) has effectively reduced high-score bias.

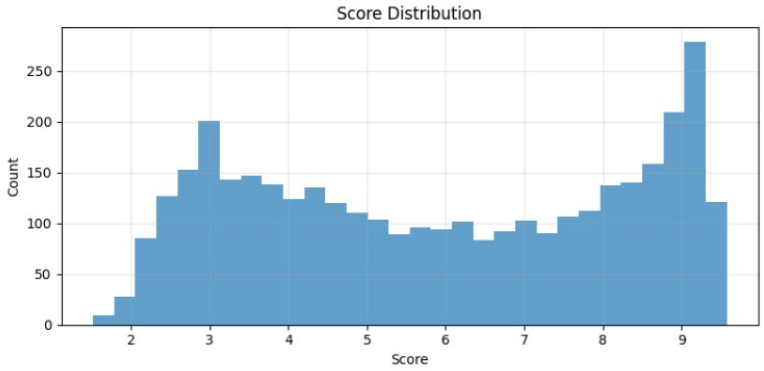

Figure 14: Distribution of predicted scores on the competition test set. Although ground-truth labels are unavailable, the distribution provides qualitative evidence of balanced test-time behavior.

6.5 Interpretation Without Ground-Truth Labels

The predictive distribution serves as the sole observable signal about the model's generalization characteristics. The alignment between the training-set distribution and the test-set predictions suggests that:

- the model avoids mode collapse,
- the regressor generalizes smoothly across the expanded training range,
- the augmentation strategy successfully prevents overconfidence toward high scores,
- prediction behavior remains stable across various metric families, as reflected in the continuity and multi-modal nature of the score density.

Ultimately, the official evaluation metric (RMSE) is computed exclusively by the Kaggle platform upon submission. The testing pipeline therefore emphasizes rigorous consistency, reproducibility, and robust distributional behavior, allowing the leaderboard score to reflect the true generalization capacity of the trained model.