DA5401 – Evaluation Metric Prediction Challenge

Kaggle Competition Report

Name: Karan Kishore Roll Number: DA25D400

The question is to develop a metric learning model to predict a (continuous) score (0 - 10) for each provided example `(system_prompt, user_prompt, response, metric_name)`. A gotcha here is that the given dataset is highly skewed towards the high score examples resulting in strong class imbalance. The dataset also has multiple languages so a simple embedding cannot work a proper multilingual embedding needs to be used. The final model combined (i) pretrained LaBSE text embeddings, (ii) cosine similarity–based topic divergence, and (iii) a regression head with tailored post-processing.

The final approach resulted in **RMSE** **2.026** in the complete test dataset.

Contents

1 Data Engineering

1.1 Dataset Description

The original training data consisted of 5,000 rows with the following fields:

- `metric_name`
- `system_prompt`
- `user_prompt`
- `response`
- `score`

There were 3,638 rows in test set without the scores but has all the other fields.

A strong imbalance is observed among the scores:

$$\text{Mean score} \approx 9.0, \quad \text{Std} < 1.0,$$

with 9 and 10 dominating the distribution. Scores below 5 are very rare. For instance, there is only one row with score as 5. In such a case, predicting score 5 definitely requires more examples as it has only 1/5000 chance of getting a score 5 among the 10 classes.

1.2 Preprocessing

All three text fields were combined into a single `full_text` field (this also feels intuitive to do so):

$$\text{full\_text} = \text{system\_prompt} + \text{user\_prompt} + \text{response}.$$

For rows with null values, an empty string "" is used to avoid NaN values.

Each row was assigned a stable `orig_index` to ensure later re-alignment of embeddings (important after augmentation).

Metric definitions are also embedded through the provided file `metric_name_embeddings.npy` with the same embedding tool as the one used for the prompts and response.

2 Sampling Strategies that were tried out

2.1 Initial Attempts: Random Oversampling / Undersampling of low classes

Multiple rounds of oversampling rare labels (0–5) and undersampling majority (9–10) are done but ironically such attempts just resulted in degraded performace. No matter what sampling is used, the RMSE didn't drop below RMSE of 3.655. I then tried experimenting on metric names and embeddings and how they are clustered and so on. One thing was that there are many clusters formed when I used T-SNE's first two dimensions. So, I thought that I could just replace the embeddings with a random embedding and set the score to a low value between 1 and 4 to augment the low class sample and shift the class distribution as this is much faster compared to using LLM to generate synthetic data or other sampling techiqnues like CBO.

2.2 Negative Sampling through Metric Shuffling

The thought in previous discussion led me to the below idea:

1. Select only rows where score ≥ 8 (high-quality responses).
2. Replace their `metric_name` with a random wrong metric.
3. Assign random integer score between 1 to 4 for these synthetic samples. (I also tried setting score = 0 which worked well (RMSE 2.6) but random integer between 1 and 4 pushed it down further).

This constructs examples where the text is certainly good but with respect to the new metric it is not relevant at all. The model learns a strong penalty for "metric mismatch" which can be useful rather than just converging to the most frequent scores through regression.

This step immediately reduced RMSE from $3.6 to 2.6$.

2.3 Topic Divergence

Then, I has the idea where I wanted to check if the scores are somehow dependent on whether the topic in prompt is similar to the topic in the

response. This involved quantifying the semantic topic alignment between the prompt and response.

$$\text{divergence} = 1 - \cos(\text{Embed}(\text{prompt}), \text{Embed}(\text{response})).$$

High divergence literally means that the response does not address the user prompt adn results in topic divergence, which by the way is a metric in itself.

This concept was useful because:

- Several metrics in metric names like coherence, refusal correctness, topic drift implicitly depend on similarity between the prompt and response pair.
- An experiment was done by just using the topic divergence and doing a prediction on that gave an RMSE of 2.61 wihtout any embedding or metric name! I felt this quite surprising leading me to the idea that I need to add some similiarity alignment in final model.

A standalone divergence-only regressor achieved RMSE ≈ 2.6, surprisingly me asno metric information was used. So, I thought about the idea of adding this as a feature (feature engineerring) which worked wonders in the best model.

3 Exploratory Data Analysis

3.1 Score Distribution

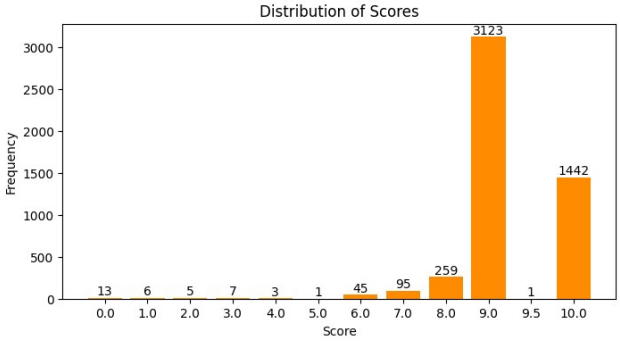

Figure 1: Score Distribution

The training data scores show a very concentrated distribution around 9–10 and sparse data for lower scores as discussed before.

3.2 Metric Frequency

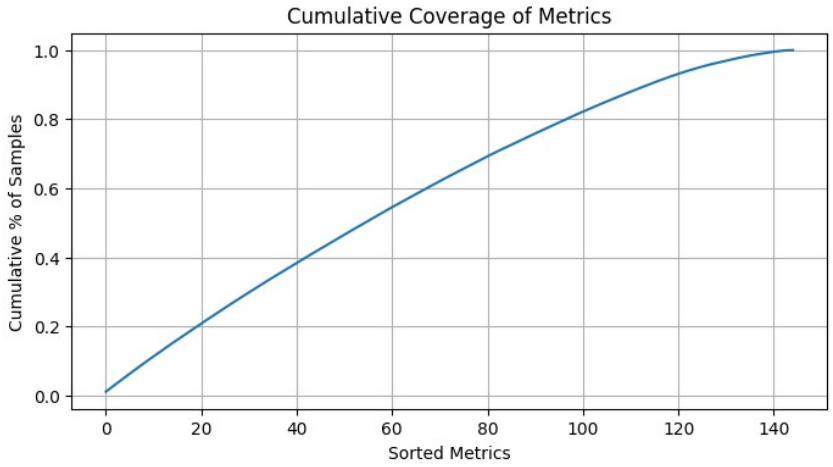

Figure 2: Coverage of metric name across all training samples to check whether a particular metric is too frequent

We can see a smooth curve showing that there is a fair distribution of the metrics among the training samples and not skewed like the scores.

3.3 Embedding Visualization

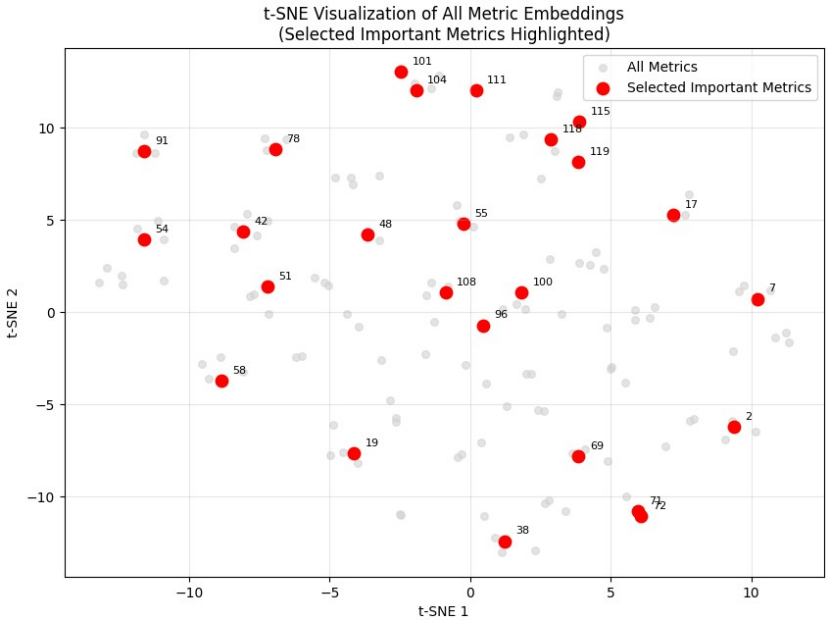

Figure 3: t-SNE 2-D projection of LaBSE embeddings of the text

3.4 Score Distribution

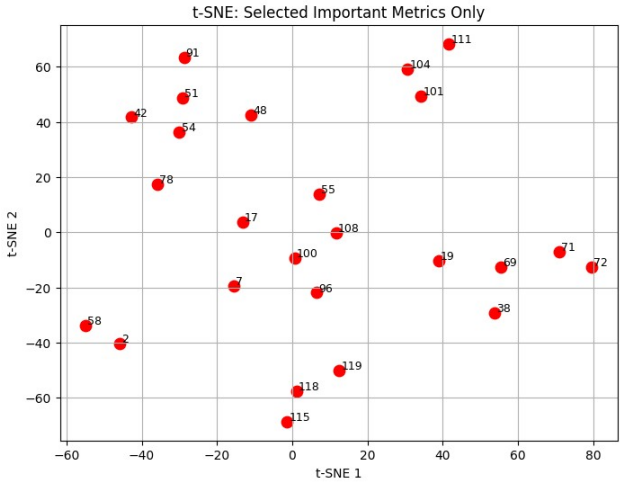

Figure 4: T-SNE embeddings for important metrics

A lot of tiny clusters emerge based on the semantic meaning of the metric name using their embeddings. Important metrics are based on the farthest distance. A detailed report on the clusters formed by considering important embeddings is as follows: The above embeddings make sense and the metrics are varied in different ways.

1. Safety and Harm Prevention

- **42**: toxicity_level
- **51**: harmful_inputs / prompt_injection
- **91**: jailbreak
- **58**: bias_detection
- **54**: inappropriate_content

2. Data Protection and Privacy

- **101**: privacy_leakage
- **104**: privacy_leakage / indirect_inference_of_private_data
- **111**: data_confidentiality

3. Robustness and Reasoning Ability

- **17**: topic_drift_rate / failure_to_recover_to_main_topic
- **7**: transparency / clarity_of_reasoning
- **55**: hallucination_rate / factual_hallucination
- **100**: anthropomorphization_response_tendency
- **108**: confidence_agreement / underconfident_correctness
- **96**: awareness_query

4. Stereotype and Bias

- **58**: bias_detection / stereotype_expression
- **2**: inclusivity / demographic_inclusivity

5. Multilingual Handling and Coherence

- **19**: dialogue_coherence / intra_turn_coherence
- **38**: fluency_score / morphological_accuracy
- **69**: ability_to_handle_multiple_indian_languages
- **71**: transliterated_language_handling / romanized_input
- **72**: transliterated_language_handling / tolerance_to_spelling_variants

6. Controlled Data Handling

- **115**: data_minimization
- **118**: data_quality_assessment
- **119**: data_governance_policies

4 Model Selection

4.1 Models tried during the competition

: I have tried the below models during the competition

- LightGBM regression on raw embeddings with class balancing using inverse frequency of classes
- LightGBM classification (treated score as 11-class problem)
- Siamese networks (rejected due to assignment constraints asking only pure ML methods)
- Gradient boosting over metric embeddings

- Ridge regression - has random score predictions across all values with RMSE of 3.2. It was not robust enough.
- Ensembles of multiple embedding families (LaBSE, MPNet, MiniLM) - still converged to suboptimal pair of scores
- using augmented dataset with samples generated using GPT-5.1 - Improved all the RMSE but by a low value around 0.08 RMSE reduction only.

Many approaches produced collapsed predictions:

$$\hat{y} \in \{6, 7\}, \quad \text{or} \quad \hat{y} \in \{9, 10\}.$$

This prediction was given no matter what sample is passed as an input to it.

In classification, I basically took 11 classification questions: Is score ¿ 1, score ¿ 2 and so on to get 11 probabilities for classes 0 to 10. Then find the place where the probability drops and set that as the score. This method also didn't work as the dataset was skewed. However it pushed the predictions from 9 or 10 to 6 or 7 taking RMSE ¿ 4 to around 3.612. Binary log loss was used in above case.

4.2 The Best Method

The final strategy combined the hacks and learnings from the failed models.

(1) Metric mismatch must definitely be penalised by assigning scores between 1 to 4. Most errors came from predicting high scores for wrong metrics in test data. Negative sampling fixed this to a very good extent.

(2) Topic divergence is a strong proxy for meaningfulness between prompt and the response. Metrics related to coherence, relevance, safety, and hallucination all correlate with prompt–response alignment.

(3) Cosine similarity behaves smoothly. Unlike regression on raw embeddings, a single scalar divergence is stable, well-behaved, and resistant to overfitting.

Thus the final formula:

$$\hat{y} = \alpha \cdot f_{\text{LGBM}}(X_{\text{LaBSE + metric}}) + (1 - \alpha) \cdot f_{\text{divergence}}.$$

With α around 0.65. The combination of:

1. LightGBM regressor with tuned tree-depth, leaf count, and
2. synthetic negatives + cosine-similarity features

proved to be the most stable and least biased This method reduced RMSE from 2.6 to around 2.02.

The final model was trained with the following parameters:

- **Learning Rate:** 0.04
- **Num Leaves:** 127
- **Depth:** 8

4. Final Feature Matrix Construction

Each training example was represented by:

- 300-dimensional metric embedding
- 768-dimensional LaBSE embedding of system prompt
- 768-dimensional LaBSE embedding of user prompt
- 768-dimensional LaBSE embedding of model response
- Three cosine similarity values

5 Hacks and Workarounds

The RMSE was not reducing below 2.6 despite dataset augmentation so following strategies were used to bring it down to 2:

- **Metric-shuffled low scoring** This was the most important workaround by replacing the metric of high scoring samples with random metrics. (I tried with explicit conflict map of opposing metric but random shuffling performed better in terms of lower RMSE scores).

- **Topic divergence feature** based on cosine similarity between the prompt and response.
- **Re-weighting and clipping predictions** to ensure valid range during final prediction. Also using a regressor as continuous values often have a lower RMSE than rounded values.

5.1 Training and Validation Performance

The final LightGBM regressor was trained using 3500 boosting rounds, determined via early stopping(100 as patience).

Iteration	**Train RMSE**	**Validation RMSE**
150	1.59247	2.59867
300	0.949989	2.47430
450	0.633436	2.45177
600	0.469708	2.44452
Best (577)	**0.489101**	**2.44332**

Table 1: RMSE progression during training of the final model.

The results show a steady decrease in both training and validation errors. Although the model continued to improve on the training set, the validation RMSE plateaued around 2.44. The test RMSE was around 2.02.