DA5401: Endsem Data Challenge
Project Report

Name:	**Devashish Tripathi**
Roll No.:	DA25C006
Date of Submission:	November 20, 2025

1 Introduction

Conversational AIs have been finding use in day-to-day life, offering assistance, advice, and dialogue across various domains. With the support for various languages increasing day by day, they are being integrated into daily life. As such, it becomes necessary to measure their performance, particularly on the basis of how well they perform on the task they are designed for. This can be measured by comparing the prompt given to the AI agent on the particular task and how close its response is to the actual prompt. This is necessary to create improved task-specific datasets and thus, conversational AI agents that can be task-specific and offer support in various languages.

Metric learning is a type of machine learning that focuses on learning a distance function to measure how similar or different objects are from each other [1]. For this purpose, this Data Challenge [2] focuses on an AI evaluation system that tests a target conversation AI by using test prompts for various evaluation metrics and assessing their closeness to the response received. The closeness is assessed on a score ranging from 1-10, with the task for this project being to construct a model in a metric-learning context, focused on learning the apt feature space connecting the prompt-response pair and how close they are to the evaluation metric.

The data consists of prompt-response pairs in a multilingual fashion, consisting of various languages, such as Tamil, English, Hindi, Bengali, Bodo, Assamese, and Sindhi. For the train data, the LLM scores are also provided, discretized in the range of 1-10. The evaluation of the model was done using RMSE, comparing the predicted score to the LLM score. As such, the goal can be formally stated as:

Given a metric definition (text embedding) and a prompt-response pair (text), predict the relevance or fitness on a scale of 1-10.

The code, relevant datasets, and weights can be found at the GitHub repository.

2 Dataset

The dataset contains prompt-language pairs in multiple Indian languages such as Tamil, Hindi, Assamese, Bengali, Bodo, Sindhi, and English. The prompts and responses are provided, along with the evaluation metric and the system prompt if present. Additionally, the embeddings of the metric names, extracted using Gemma-300M [3], were also provided.

As the dataset had scores discretized in the [0, 10] interval, they were counted, with their counts in Table 1.

Over 90% of the samples were observed from the score of 8-10, with lower scores almost completely absent and the middle-range scores of 6-7 exceedingly rare, making direct regression extremely difficult due to the training data's bias towards high-valued scores.

The language distribution in the dataset was also evaluated, necessary to select a suitable language model for embedding extraction from the text. For this purpose, the `fasttext` [[4]; [5]] library was utilised. The language distribution on the user prompts can be seen in Figure 1.

Score	Count
10.0	1442
9.5	1
9.0	3123
8.0	259
7.0	95
6.0	45
5.0	1
4.0	3
3.0	7
2.0	5
1.0	6
0.0	13

Table 1: Table showcasing the counts of occurrences in the dataset

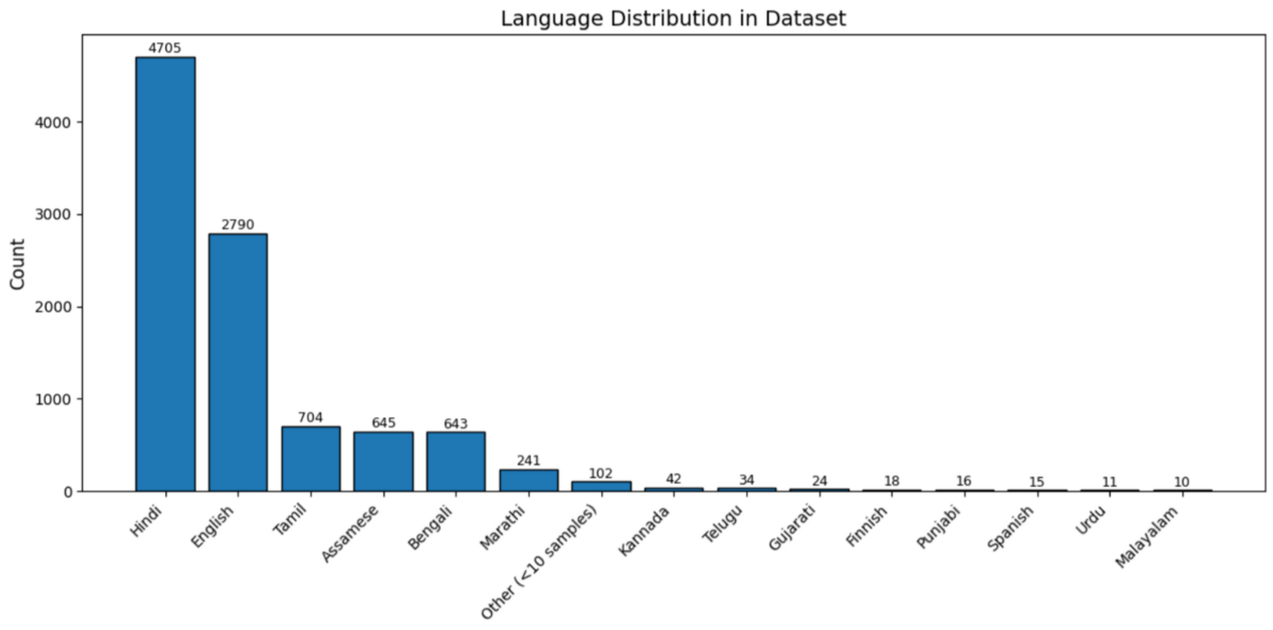

Figure 1: Bar Plot illustrating the language distribution on the user prompts in the dataset

As can be seen, Hindi has a significantly high appearance, even surpassing English and Tamil. However, this might also be due to the use of common scripts (Bodo and Hindi both use Devanagari) and the model not recognizing Indic languages well. Either way, the distribution showcases heavy Indic focus, prompting the need for a language model (LM) catering to the same.

Null values were also present in the dataset, with there being one `response` and several `system prompt` columns being empty. In order to tackle these, and generalise, if any column with some column name `cname` was encountered having null values, a default value of `<NO cname PROVIDED>` was imputed.

3 Feature Engineering

For such a task, feature design was the core purpose, and various feature design methods were incorporated, each iteration incorporating aspects of the previous based on model performance, evaluated through train and test performance (through the proxy of the challenge leaderboard).

3.1 LM-Based Embedding Features

Based on the language distribution, it was apparent that models with indic focus were to be considered. As such, four embedding models were explored:

1. Vyakyarth [6]: A Multilingual model for Indic Languages, was chosen for cross-lingual performance.
2. Multilingual-E5-base [7]: Used to provide semantic representation to the less represented Indic languages such as Bodo, which might be otherwise confused with other languages
3. IndicBertv2-MLM [8]: Offers support for 23 Indic languages, superseding Vyakyarth's 10 language support.
4. Google Embedding-Gemma (300M) [3]: Used to align the metric embedding space closer to the prompt-response embedding space.

After experimentation, the most consistent and robust performance came from IndicBERT, followed by Gemma, Vyakyarth, and Multilingual-E5-base. As such, for the final model, the embeddings were weighed accordingly when used for feature extraction, with their respective contribution being 2, 1, 0.5, and 0.25. All models output 768-dimensional embeddings, allowing their combinations to be made relatively easily. The embeddings were combined in a linear fashion with their respective contributions as outlined previously. The features were loaded using the `transformers` [9] library.

3.2 TF-IDF-based Statistical Embeddings

While LM-based features capture semantic meaning, they might miss information such as repeated patterns, verbosity, or the aptness of terms. As such, statistical methods like those of TF-IDF help by emphasising such text-based information, such as counts, differences, and phrases, adding interpretability to the features. High TF-IDF weights indicate important but rare information, while lower weights mean less importance.

3.3 Stagewise Feature Engineering

Feature extraction from the embeddings extracted in the previous two methods was done as follows:

3.3.1 Stage 1: Pairwise cosine similarity features

For initial experimentation, pairwise cosine similarity was captured and used as the model's features. Cosine similarities were captured between the following pairs:

- `user_prompt` and `response`: The key factor measured by the LLM judge is their closeness
- `system_prompt` and `response`: To see if the response aligns with the `system_prompt`, showcasing how good the conversation agent is at following commands
- `metric` and `response`: To see if the response fits the metric well, showcasing if the conversation agent understands the task well
- `metric` and `user_prompt`: To see how close the user prompt is to the intended task, showcasing how much the agent penalised (or rewarded) even if the user prompt was not correct

This resulted in 4 similarity features.

3.3.2 Stage 2: TF-IDF Features

Based on the results on just the Stage 1 features, TF-IDF features were introduced. User Prompt and response text pairs were concatenated, and 25 TF-IDF features were calculated using `scikit-learn`'s [10] `TfidfVectorizer` module for unigrams-trigrams. This resulted in 29 features.

3.3.3 Stage 3

Following the results post Stages 1 and 2, it was evident that more information was to be extracted from the dataset to achieve an improvement in the performance. As such, the combined mean of the user prompt, response, and system prompt embeddings was also taken as a 768-dimensional feature vector, and two metrics were also extracted from them, which were the difference between the mean embeddings and the metric embeddings, measuring the difference between how much the performance differs from the metric, and the hadamard similarity was also used to capture the same. This resulted in 770 features.

3.3.4 Stage 4

Due to the number of features, it became necessary to use dimensionality reduction in order to avoid overfitting. PCA was applied in three ways: Only on the Stage 3 results, on the combined Stage 1 and Stage 3 results, and on the PCA-Stage 3 results and Stage 1 results. All of these were done at various explained variance ratios. Based on experimentation, the best results came when the features of the three stages were combined, and then PCA was done at a 0.99 explained variance ratio, leading to the final data having 533 features.

4 Modelling Approaches and Results

Initial experimentation was done on just the Stage-1 features. GridSearch was employed to do a parameter sweep across various hyperparameters. Experimentation was done on Linear Models (such as Ridge, Lasso, or ElasticNet), KNN, SVM, Random Forest, XGBoost [11], LightGBM, and MLP-based Regressors. While the train RMSE was between 0.94 and 1.00, the test RMSE stayed at 4.78 for the best models, which were Random Forest, XGBoost, and MLP-Regressor, chosen based on training time and training RMSE.

Additional features were added following Stage-2, which increased the train RMSE to 3.5, however, pulled down the test RMSE marginally to 4.25, which is a possible indication of the fact that the available test data for the public leaderboard was skewed towards the middle-range scores. This step, however, solidified the Random Forest Regressor as the model of the choice.

Considering this fact, more features were now tested, including Mahalanobis, to varying degrees. Finally, Stage-3 features were incorporated, and the test RMSE lowered to 3.93. A two-stage model was also employed, where a classifier would first predict whether the datapoint is a rare or a LightGBM [12] classifier was used for the same. Two separate regressors were trained for the same. No significant improvement was noticed, with performance degrading when Stage-features were introduced, leading to the dropping of this approach.

Finally, once Stage-4 features were incorporated, the model's performance improved significantly, with the train results improving to an RMSE of 2.48, while the test results fell to an RMSE of 3.690, indicating that the current feature representation was providing adequate representation.

5 Discussions and Conclusions

Given the distribution between the train and test sets, it was a daunting task to get an RMSE under 4. The unbalanced train data also caused significant issues, leading to modeling approaches focusing on mitigating the issues caused by the same. In this task case, an RMSE under 2 might be considered ideal. Though my approach was not able to do the same, there were certain insights gained, which could be used in the future.

Purely embedding-based features might not always work due to significant overlap in the feature space, particularly in heavy imbalance and distribution change cases such as these. As such, statistical features are always welcome.

Among various strategies tried and failed, such as oversampling, classification-regression pipeline, undersampling, and Mahalanobis centroids, several insights were gained. The rare label samples were exceedingly few, meaning any upsampling would mean noise introduction, while downsampling the frequent label samples also meant that information on certain evaluation metrics might be lost, and thus, performance might take a hit on the entire data. The classification-regression pipeline was thresholded and parameter-tuned, but needed more data to build a more robust classifier. As such, if evaluation-metric free sampling of the frequent labels might have helped in the same. Mahalanobis features not providing any improvements shows that different evaluation metrics would have had different spaces, and as such, this metric, which works on the distance between samples and distributions, was not a suitable fit here, and there might be little covariance in the features.

As such, despite not acheiving the best possible results, the project shows how structured modeling, along with intelligent feature extraction using domain knowledge can be useful in metric-learning based problems, even in cases of skewed datasets and different distributions between the train and the test sets. The final model achieved a train RMSE of ≈ 2.48 and a test RMSE on the public leaderboard of 3.690.

Future work can be focused on improving the modeling, such as trying out more tuning in the classification-regression pipeline, using methods such as GMM to create synthetic samples, and adding more statistical features, such as bag-of-words.

References

[1] Aurélien Bellet, Amaury Habrard, and Marc Sebban. *Metric Learning*. Synthesis lectures on artificial intelligence and machine learning. Springer International Publishing, Cham, 2015.

[2] Sudarsun Santhiappan. Da5401-2025-data-challenge. https://kaggle.com/competitions/da5401-2025-data-challenge, 2025. Kaggle.

[3] Henrique* Schechter Vera, Sahil* Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, ... Naim, and Mojtaba Seyedhosseini. Embedding gemma: Powerful and lightweight text representations, 2025.

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[5] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[6] Rajkiran Panuganti Pushkar Singh, Sandeep Kumar Pandey. Vyakyarth: A multilingual sentence embedding model for indic languages. https://github.com/ola-krutrim/Vyakyarth, 2024.

[7] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

[8] Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada, July 2023. Association for Computational Linguistics.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. ACM, August 2016.

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.