DA5401 Kaggle Challenge : Metric Learning

G C V Sairam
Roll No : DA25M012

November 20, 2025

1 Introduction

1.1 Problem Statement

There is an AI evaluation system which tests a target conversational AI agent with a variety of test prompts which includes many use cases from different business domains. The evaluation process takes the test prompts and their responses and assesses them on a variety of evaluation metrics. We are given a dataset of 3 components : Metric Definitions, Prompt-Response Pairs, and target fitness scores which represent the score given by the LLM judge. Our task is to train a model that takes the metric definitions and prompt-response pairs as inputs and predicts the fitness score in a range of 0-10. So, the model must learn the semantic distance between 2 pieces of text that describe the intent and the test case.

1.2 Dataset

We are given a dataset with 5000 training data points and 3638 test data points. Each datapoint has two types of embeddings: one is the metric definition and the other represents the combined text containing the system prompt, user message, and the response. The target scores ranged from 0 to 10 but were extremely skewed toward higher values 9 and 10.

2 Methodology

2.1 Data Pre-processing

Creating embeddings

In the dataset, there are three text fields: system prompt, user message, and the model's response. To make the embedding model handle the entire context consistently, I merged these into a single block of text. Both the combined text and the metric name were then converted into 768-dimensional numerical embeddings using the `l3cube-pune/indic-sentence-bert-nli` model which is a very good model for Indian languages.

Handling the score imbalance

Because the data was heavily imbalanced towards higher scores, I computed sample weights based on inverse frequency of each output score, so that the model does not completely ignore the lower scores.

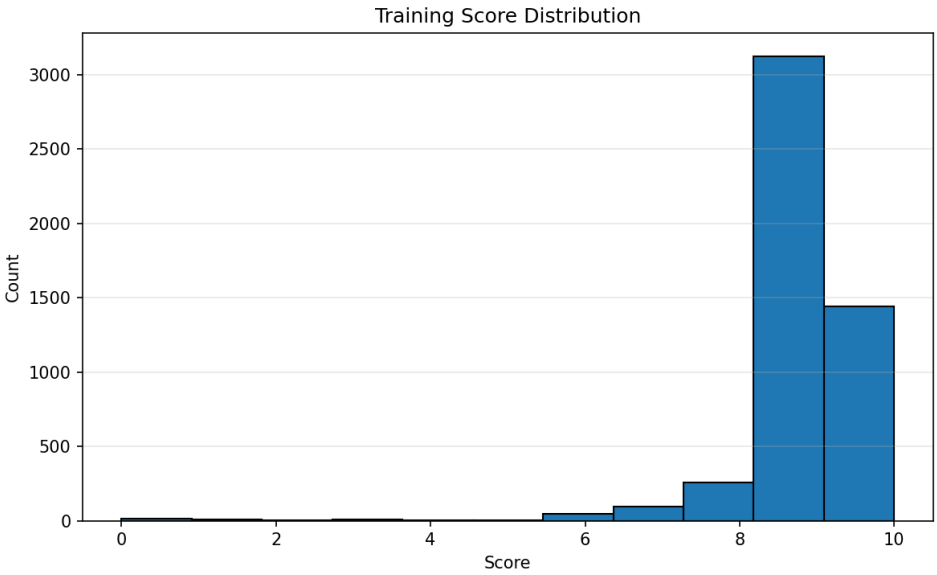

Figure 1: Score distribution in the training set.

2.2 Model Architecture

We designed a dual-tower neural network. One tower took as input the metric embedding, while the other took as input the prompt–response embedding. In each tower, there is a Dense layer with a ReLU activation function.This was followed by dropout and L2 regularization to avoid overfitting. After the towers processed their respective inputs, I concatenated the tower outputs into a single 1024-dimensional vector, which was then passed through another Dense layer with a linear activation function to output the predicted fitness score.

By doing this, we allow for the model to learn the individual representations for both the inputs before combining them.

2.3 Model Complexity

Each of the 2 towers has almost 400,000 parameters. After dropout is applied, the outputs of both towers are joined into a single 1024-dimensional representation. The final prediction layer is a simple linear transformation, which adds around 1000 more additional parameters. In total, the entire network has a little under eight hundred thousand trainable parameters.

2.4 Training the model

The model was trained using the Adam optimizer with a low learning rate.For the loss function, I used the mean squared error function and for the evaluation metric, I used the root mean squared error (RMSE) during training. Maximum number of epochs was set to be 40. In each epoch, Ten percent of the training data was set aside for validation, and early stopping with patience value = 5 was used to avoid unnecessary epochs.

Over the course of training, both training and validation errors dropped steadily. By the time training stopped, the validation RMSE was around 3.9.

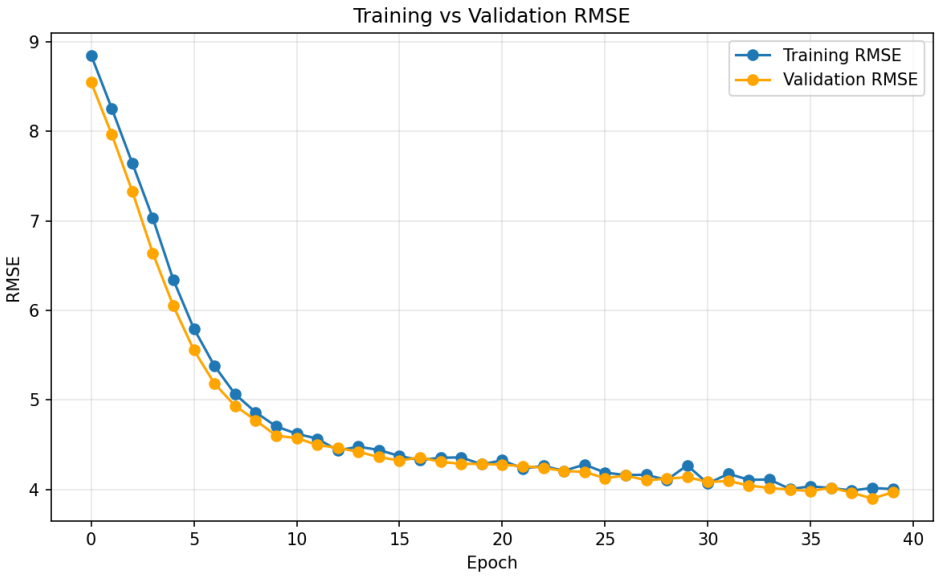

Figure 2: Training vs. validation RMSE across epochs.

3 Results

3.1 Training Error

In training, both the training and validation RMSE steadily decreased. By the end, the validation RMSE was approximately 3.9.

3.2 Observations on the Test Predictions

When applying the model on the test set, the predictions clustered mostly around scores of 5 and 6. Only a very small number of predictions were outside this range. This was completely different from the training data distribution which was heavily skewed towards higher scores 9 and 10.

This could be due to the very high imbalance in the training data due to which the model could not identify the differences between high-scoring datapoints and low-scoring datapoints that well. So the model just gave the output as the midpoint of the 0-10 range for most of the datapoints , because of which we are getting an RMSE of 3.68 .

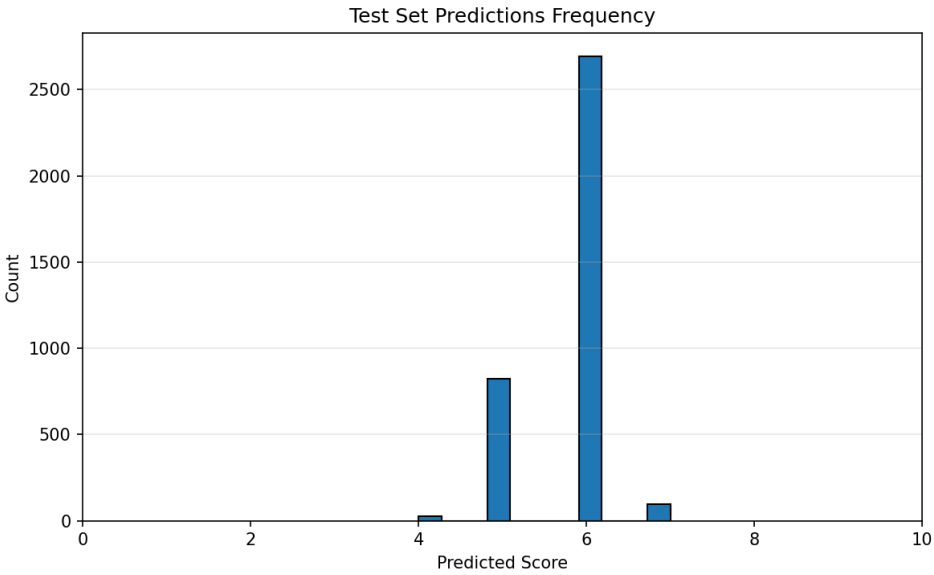

Figure 3: Predictions on Test Samples.

4 Pros and Cons of the model

One advantage of this approach is that it uses embeddings that already encode much of the semantic meaning. This keeps the model small and easy to train. The dual-input structure also makes sense for a task where two different textual sources need to be compared.

but, the limited range of test predictions shows that the model still did not learn to actually judge the extremes of the scoring spectrum. Predictions were mostly in the 5-6 range instead of covering the entire 0-10 range.

5 Conclusion

We implemented a relatively simple neural model to be used to estimate AI evaluation scores based on metric embeddings and prompt-response text embeddings. The model trained stably and learned some useful patterns, although its predictions were mostly in the mid-range.