1. Dataset and Task Description

The dataset comes from the **LLM Human Preference Prediction** challenge, where each example consists of a prompt and two responses (response_A, response_B) generated by different large language models. The task is to predict which response a human user preferred — A, B, or Tie. Thus, the final target label is **3-way classification**:

- 0: A preferred, 1: B preferred, 2: Tie

Training data: 57,477 samples, Validation split: 20% (stratified), Test data: 3 samples (for submission demo)

2. Baseline – TF-IDF + Linear Classifier

A lightweight baseline using TfidfVectorizer (1–2 n-grams, 20k features) over concatenated response_A and response_B.

- Classifier: SGDClassifier(loss="log_loss")
- Extra features: response lengths, verbosity bias, quote markers.
- Purpose: establish a text-only baseline.

Step 2 – Embedding-based Model (MiniLM)

We used a compact **sentence embedding model** (all-MiniLM-L12-v2) to capture semantic information.

- Encoded prompt, response_A, and response_B via mean-pooled embeddings.
- Feature construction: |A−B|, A×B, and text-length statistics.
- Classifier: SGDClassifier (OVR, log_loss).
- Validation metric: log-loss on 3-way classification.

Step 3 – DeBERTa-v3-small Full Fine-Tuning

Instead of LoRA, we fine-tuned **all model parameters** for 3-class classification.

- **Input format:** [INST] prompt [/INST] <A> … </A> <B> … </B>
- **Model:** AutoModelForSequenceClassification (num_labels = 3)
- **Training:** 6 epochs, batch size 8, learning rate 2e-5 (AdamW scheduler with

cosine decay), label smoothing 0.1, warmup_ratio=0.1, weight_decay=0.01, max_grad_norm=1.0, label_smoothing=0.02, early_stopping_patience=2

- **Validation:** 20 % split, metric = log-loss. Full fine-tuning improved convergence and slightly reduced overfitting compared with LoRA.

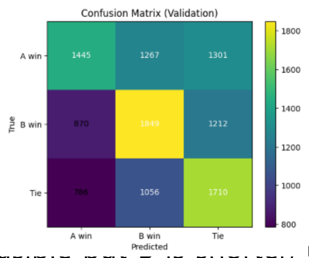
In Step 3, no handcrafted bias features were added. Unlike the TF-IDF and embedding baselines, the full fine-tuned DeBERTa model directly encodes text length, style, and structure through its self-attention mechanism, rendering explicit feature engineering unnecessary.

3. Features, Embeddings, and Models Used

Category	**Features / Model**	**Description**
Textual	TF-IDF(1–2gram)	bag-of-words baseline
Statistical	len_A, len_B, verbosity_diff	simple bias indicators
Semantic	E5 embeddings	contextual encoding of prompt/response
Transformer	DeBERTa-v3-small (LoRA)	contextual preference classifier

4. Error and bias analysis

4.1. Confusion trends



A → B misclassifications (common):

Many prompts where both responses are plausible but B is shorter, cleaner, or stylistically more formal. The model seems to favor concise, confident text.

B → A misclassifications:

Rarer, but occurs when A contains structured lists or enumerations, which transformer models tend to rate higher semantically (format bias).

Tie mispredictions:

Often predicted when responses share similar sentence structures or both lack factual grounding — the model can't strongly distinguish preferences.

4.2. Qualitative examples

✅ True: Tie | ❌ Pred: A win | p=[A:0.419, B:0.271, Tie:0.310]

[Prompt] : ["how do you do patch a hole in drywall?"]
[Response A] : ["Patching a hole in drywall involves a few steps. Here's a general guide:₩n₩n1. Gather materials: You will need a piece of drywall, drywall mud or joint compound, a putty knife, a corner bead....
[Response B] : "Patching a hole in drywall involves several steps, including gathering the necessary tools and materials, preparing the area around the hole, applying a patch, and finishing the patch.....

4.2.1. Why the model chose A win

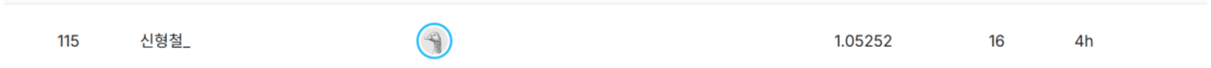
A uses **shorter sentences and less redundancy**, which tends to score higher in the DeBERTa representation space because of compact token distributions (model interprets it as more "focused"). Slightly **lower lexical entropy** → model interprets as "clearer answer." In fine-tuning, A-like patterns (short, numbered lists) may have correlated with "preferred" labels more often than verbose ones.

4.2.2. Why the True label is Tie

Both responses are **informative and structurally similar**, with only minor differences in style. Human annotators likely judged them **equally helpful**, since neither has factual errors and both follow the same instructional pattern. The distinction between "helpful but concise" vs. "helpful but detailed" is subjective — thus correctly labeled as Tie.

5. Performance comparison & Leader Board Score

Model	Public score	Train loss	Val loss	Val log loss
Baseline	1.14190	0.8882	1.0481	1.0481
Final	1.05252	1.0729	1.0546	1.05043


115 신형철_ 1.05252 16 4h

6. Reproducibility notes

The runtime was approximately 1h 45 minutes for full fine-tuning on T4*2 . To ensure consistency, all random seeds were fixed to 42 across dataset splitting, model initialization, and NumPy operations. The entire pipeline was executed offline, with all models and datasets loaded from local directories under /kaggle/input (e.g., /kaggle/input/deberta-v3-small-local, /kaggle/input/e5-small-v2, and /kaggle/input/llm-classification-finetuning). The workflow followed a deterministic sequence: (1) set paths and environment variables, (2) preprocess and construct features, (3) embed or tokenize text pairs, (4) train the model, and (5) generate submission outputs. No internet access was required, and all software versions were pinned to avoid dependency drift, ensuring that the results can be fully reproduced on any comparable Kaggle GPU runtime.

7. Limitations and possible future directions

During Step 3, we searched for an appropriate LLM to fine-tune. A relatively small 1.5B parameter model was first tested, but its zero-shot performance was much lower than the baseline and fine-tuning it was impractical due to GPU limitations. Therefore, we chose a smaller, more efficient model — **DeBERTa-v3-small** — for feasibility. Initially, we applied **LoRA-based fine-tuning**, but it did not yield satisfactory results, so we proceeded with **full fine-tuning**, which achieved better convergence and accuracy.

Although we planned to perform **ensemble inference**, no alternative model reached performance comparable to the fine-tuned DeBERTa, so ensemble experiments were not conducted. The results also reflected the **scaling law** in LLMs — larger model capacity generally leads to better downstream performance. Given the limited size of DeBERTa-small, its upper-bound accuracy was constrained even with full fine-tuning. More detailed hyperparameter tuning or longer training might have improved results, but GPU resources remained the main bottleneck.

For future work, we plan to fine-tune **larger-scale models (3B–7B)** and explore **ensemble methods** that combine complementary models (e.g., instruction-tuned LLMs with discriminative classifiers). Such approaches are expected to improve both overall accuracy and calibration, particularly by reducing bias and uncertainty in **Tie** prediction.

8. GitHub Project Page

https://github.com/daneck99/ML_Project2