Team Project 2

Team 7

Github: LINK: https://github.com/junseoyou/MachineLearningProject_TeamProj2_Team7

1. Description of Dataset and Task

Task: aggle competition **LLM Classification Finetuning**. The task is a 3-class classification problem designed to predict human preferences for Large Language Model (LLM) responses.

Dataset: The dataset consists of 57,477 training samples and approximately 25,000 test samples. Each sample contains a prompt, response_a, and response_b.

2. Baseline Methods and Model Extensions

2.1. Baseline Models (Step 1 & 2)

Baseline 1 (Lexical Model): We implemented a simple lexical feature model. We created a "Simple Baseline" using 10 features (e.g., character/token counts and their differences) and trained a **LogisticRegression** classifier. This yielded a validation Log Loss of **1.070**.

Baseline 2 (Embedding Model): We tested seven pre-trained sentence embedding models (including all-MiniLM-L6-v2, L12-v2, e5-small-v2, base-v2, large-v2, sentence-t5-base, and t5-large) combined with two classifiers (Logistic Regression and MLPClassifier).
The best performing baseline was **e5-small-v2** combined with an **MLPClassifier**, achieving a validation Log Loss of **1.053**. A key part of this step was creating interaction features from the prompt, response A, and response B embeddings (e.g., A-B, A*B).

	all-MiniLM-L6-v2	all-MiniLM-L12-v2	e5-small-v2	e5-base-v2	e5-large-v2	sentence-t5-base	sentence-t5-large
mlp loss	1.072	2.105	**1.053**	1.058	2.136	1.734	2.114
Logistic loss	1.096	1.11	1.1	1.172	1.274	1.175	1.182

2.2. Model Extensions (Step 3)

We designed three distinct candidate models to form a high-performance ensemble. A key enhancement across all candidates was the addition of Isotonic Regression for **calibration**.

2.2.1. Candidate 1: Fine-tuned Transformer (DeBERTa-v3-base + LoRA)

We use a **DeBERTa-v3-base** model and apply Parameter-Efficient Fine-Tuning (PEFT) using **LoRA**. This model takes the raw text of the prompt and both responses as input and fine-tunes the transformer's attention layers to learn the nuanced task of preference classification.

2.2.2. Candidate 2: Embedding + Tree Model (e5-base-v2 + LightGBM)

To introduce model diversity, our second candidate explores **tree-based classifiers**. We implemented code to test both LightGBM (LGBM) and XGBoost using the interaction features generated from the e5-base-v2 embedding model. Both models are adept at finding sharp, rule-based decision boundaries that an MLP might miss. In our experiments, **LightGBM** demonstrated strong performance (Validation LogLoss: 1.099) and fast GPU-accelerated training, leading us to select it as our primary tree-based candidate for the ensemble.

	Validation LogLoss	Time Taken (s)
LGBM	1.0335	527.87
XGBoost	1.0330	94.80

2.2.3. Candidate 3: Hybrid Features + MLP (All-Features + MLP)

Our third candidate combines all available information. It horizontally stacks the "Strong Lexical Features" (30+ features from Baseline 1, including **bias-aware features**) with the e5-base-v2 Embedding Features. This combined feature set is then fed into an **MLPClassifier**. This model is uniquely capable of learning complex, non-linear relationships between semantic meaning (from embeddings) and surface-level statistics (from lexical features).

2.2.4. Ensemble Analysis

Our final submission is an **Ensemble** of our three candidate models. We explored five advanced ensemble strategies: Simple Average, Weighted Average (based on validation performance), Optimal Weights (via SLSQP optimization), Stacked Generalization (using Logistic Regression as a meta-learner), and Rank Averaging. The Stacked Generalization method yielded the best ensemble validation score **1.0716**.

	Simple Average	Weighted Average	Optimal Weights	**Stacked Generalization**	Rank Averaging
LogLoss	1.1065	1.1037	1.0735	**1.0716**	1.1957

2.3. Final Model (Candidate 2: e5-base-v2 + XGBoost)

Despite the sophisticated nature of our ensemble techniques, the best individual calibrated model, Candidate 2 (C2), achieved a superior validation LogLoss of 1.0301.
Reason for Selection: This score significantly outperformed the best ensemble result, indicating that the diversity introduced by the other models was detrimental rather than beneficial. For this reason, our final submission was based solely on the highly optimized Candidate 2 model, which leverages e5-base-v2 interaction features with the XGBoost.

3. Key Features, Embeddings, and Models

Our final model's success relies on three core components.

3.1. Key Embeddings and Interaction Features

Our embedding-based models (C2 and C3) are built on **e5-base-v2**. Critically, we do not just concatenate the embeddings of (P, A, B). We create 8-way interaction features [P, A, B, A-B, |A-B|, A*B, P*A, P*B] The A-B (difference) and A*B (element-wise product) features are the most crucial, as they explicitly provide the model with a relational signal for comparing the two responses.

3.2. Bias-Aware Lexical Features

To address the **bias-aware features** (position bias, verbosity) requirement, our function building stroing lexical features (used in C3) explicitly models **verbosity bias**. We engineered 30+ features, including *“d_len_char: len(A) - len(B)”, “d_len_tok: words(A) - words(B)”, and “r_len_char: (len(A)+1) / (len(B)+1)”* These features allow Candidate 3 to directly learn if humans in this dataset have a preference for longer or shorter answers, which is a known bias.

3.3. Calibration

Models like MLPs, XGBoost, and fine-tuned transformers are often over-confident, meaning their predicted probabilities do not match the true likelihood of being correct. Calibration is the process of re-scaling these probabilities to make them more reliable.
We applied Isotonic Regression to the validation predictions of each candidate model. This process significantly improved the quality of the probabilities before they were fed into the final ensemble.

	Can 1 (DeBERTa)	Can 2 (XGBoost)	Can3 (MLP)
Before Cali	1.0988	1.0358	1.0489
After Cali	1.0735	**1.0301**	1.0369

4. Validation Strategy and Results

4.1. Validation Strategy

All models were validated using a consistent strategy:

Split: A single train_test_split was used, reserving **20%** of the training data as a validation set.

Stratification: We used stratify=y to ensure the 80/20 split maintained the original class distribution of (A win, B win, Tie).

Seed: A seed was used for all splits and model initializations to ensure reproducible results.

4.2. Performance Comparison Table

The following table summarizes the validation Log Loss for each model component.

Model Description	Key Technique	Validation Log Loss
Random Guess	[0.33, 0.33, 0.33]	1.0986
Baseline 1 (Simple)	10 Lexical Features + Logistic Regression	1.070
Baseline 2 (Best)	e5-small-v2 + Interaction Feats + MLP	1.053
Candidate 1	DeBERTa-v3-base + LoRA + Calibration	*1.0735*
Candidate 2	e5-base-v2 + Interaction Feats + LightGBM + Calibration	***1.0301***
Candidate 3	All-Features (Lexical + Embedding) + MLP + Calibration	*1.0369*
Ensemble Model	**Ensemble (C1 + C2 + C3)**	1.0716
Final Model (Same with Can2)	e5-base-v2 + Interaction Feats + LightGBM + Calibration	***1.0301***

5. Kaggle Leaderboard Score

Our final ensemble model achieved a Public Leaderboard Log Loss score of **[Your Final Score]**.

6. Error and Bias Analysis (Step 4)

Bias Analysis: As described in Section 3.2, "verbosity bias" was explicitly modeled as a feature in Candidate 3. Our function building lexical features, allowing the model to learn and leverage any systematic human preference for **longer** answers.

Qualitative Error Analysis:

Ensemble Inefficiency: Our best ensemble underperformed our best individual model We ultimately chose C2 as the final model due to its superior validation score.

Ambiguous Ties: All models struggled most with differentiating a clear winner when responses were of similar quality, leading to confusion between a “tie” and a narrow win.

Candidate 2 (XGBoost) Errors: This model, being rule-based on features, failed when two responses were lexically similar but one had a subtle semantic flaw (e.g., a factual error) that the e5-base embeddings did not fully capture.

7. Reproducibility Notes

Environment: The solution requires Python 3.10 and key libraries specified in *requirements.txt*, including transformers, peft, lightgbm, sklearn, and torch.

Workflow:

- **Local Training:** All candidate models (C1, C2, C3) are trained on a local machine with a GPU using train.csv.
- **Kaggle Inference:** The local /models is uploaded as a private Kaggle Dataset. The inference notebook is run on Kaggle with Internet: OFF. It loads the competition test.csv and our pre-trained models from the Kaggle Dataset. It generates predictions for C1, C2, and C3, ensembles them, and saves the final submission.csv.

Seeds: *random_state=20010815* is used for all data splits and model initializations.

8. Limitations and Future Directions

Limitations:

- **Ensemble Inefficiency**: Our best ensemble (Stacked, 1.072) underperformed our best individual model (C2-XGBoost, 1.030).
- **Shallow C1 Training:** Candidate 1 (DeBERTa) was fine-tuned with LoRA for only 2 epochs. However, extensive epochs in fine-tuning often lead to overfitting. The balance between training depth and generalization was not achieved, resulting in this model exhibiting the worst performance among the candidates.
- **Token Length Truncation:** Candidate 1 (DeBERTa) has a token limit of 512. To handle longer inputs, we set *truncation=True*. This was not a trivial issue; our analysis shows that 57.48% of the training samples exceeded this limit. This means a significant portion of our data lost potentially valuable information from the end of the prompts or responses, which could have impacted the model's fine-tuning performance.

Future Direction:

- **Optimize the Winner**: Given its clear superiority, future work should focus entirely on Candidate 2 (e5 + XGBoost). We could experiment with different embedding models (like e5-large) or tune the XGBoost hyperparameters more aggressively.
- **Model Decorrelation**: To build a better ensemble, we need models that fail on different types of problems. A future ensemble could combine C2 (XGBoost) with a C1 (DeBERTa) model that is specifically fine-tuned on the *errors* that C2 makes.