Efficient LLM Preference Classification Through Position Bias Mitigation and Architectural Symmetry

Vatsal Vaghasiya, Nancy Kshetrimayum,
Dr.B.Prabadevi, Dr.Boppuru Rudra Prathap
Faculty of Engineering, M.S. Ramaiah University of Applied Sciences
Email: {25ETCS115018, 25ETCS126011}@msruas.ac.in

Abstract

The rapid development of LLMs has outpaced the development of reliable and scalable methods to evaluate models on the output level according to human preferences. Currently, methods rely on large models or expensive human supervision to evaluate model output. In this paper, we present a low-resource solution that is as competitive as a DeBERTa-v3-extra-small model with a mere 71 million parameters. We tackle four challenges that arise in learning from the preference of LLMs: (i) a position bias where LLMs prefer an answer based on the position of answer tokens in a string rather than their factual correctness, (ii) a multi-turn context degradation issue caused by tokenization, (iii) a low-resource computational problem, and (iv) a probability calibration problem.

We propose a symmetric Siamese architecture with identical encoders for both paired responses, new swap augmentation to double the amount of training data and eliminate positional bias, and explicit difference features as an extension of absolute difference between response embeddings ($|e_A - e_B|$) so that the model could focus on comparative signals. Using post-hoc temperature calibration, with optimal temperature $T = 1.20$, this 20% overconfidence bias can be corrected in the raw predictions.

Our model yields a validation log loss of 0.9871, a 6.2% relative improvement over the baselines, on the LMSYS Chatbot Arena dataset consisting of 57,477 pairwise human preferences. Our model achieves a 52.06% accuracy on a three-class problem. We only need 30% of the training data to achieve 95.1% of peak performance, indicating data efficiency. These results show that these newly made models can get better performance with 127 times fewer parameters which are being used in the baseline models.

Keywords: Large Language Models, Preference Learning, Siamese Networks, Position Bias, Transfer Learning, Natural Language Processing

1 Introduction

Large Language models (LLMs) has already transformed the area of AI and natural language processing(NLP). The different models based on GPT-4, Claude, Gemini, and their open-source versions have shown their high efficiency in a number of tasks, such as answering to given questions, code-generation, creative-writing, and thought processes. With the gradual acceptance of these models in production systems and the daily applications, it has become more difficult to determine whether and how to compare their outputs. The existing automated metrics uses like BLEU, ROUGE, and perplexity scores are insufficient in evaluating the essence that distinguishes the response as better, that is, the level of helpfulness, accuracy, and the response's correspondence to the intention of the user.

The LLMs are primarily evaluated with human preference evaluation and the LMSYS Chatbot Arena is used to compare the outputs of two LLMs. Human preference data collection involves significant expenses and time, making it impossible for developers to develop and test models. Hence it is that scientists have developed automated preference prediction systems that train on few samples and replicate human choice.

There are several basic problems in LLM preference classification that distinguish the task from the standard text classification tasks. First, preference judgments are subjective and hence considerable noise is contributed to the labels since different annotators may rightfully differ over which reply is best in a prompt that has been given. Second, multiple dimensions of quality, such as factual accuracy, coherence, style, and relevance, should be assessed at one time during the evaluation. Third, standard language models have context windows that are not sufficient to accommodate the long and complex LLM outputs hence better truncation or summarization techniques should be used.

There are some critical challenges in LLM evaluation. **Position Bias** is one of them. where the evaluators and even AI models prefer the responses based on their position rather than their actual quality. Previous studies have shown that models such as GPT 4 usually tend to go with the first response, regardless of the quality of the content. This bias significantly lowers the validity of preference predictions and it must be fixed. Another challenge is **Multi-Turn Context Loss.** Modern LLM interactions usually span many conversational turns, with the quality of a response depending crucially on the previous dialogue context. Naive tokenization methods that ignore turn boundaries risk conflating distinct conversational contributions, leading to degraded representations that deteriorate to capture the full

richness of the interaction. These kind of challenges are made worse by **Computational Constraints**, while state of the art preference predictors attain impressive performance, they usually rely on models with billions of parameters, requiring considerable resources power use for computation for both training and assumption. This creates obstacles to adoption in resource-constrained environments and constraints the scalability of preference-based assessment pipelines. Neural classifiers frequently exhibit overconfidence in their predictions, generating probability calculates that methodically overstate certainty. For preference classification, **Probability Calibration** is necessary for downstream applications comprising model selection, ambiguity quantification, and threshold-based decision making.

The way Transformer architectures have developed really sets the stage for tackling these issues. The attention mechanism given by Vaswani et al. changed the game for sequence modeling, allowing for direct connections between any points in a sequence. BERT showed how effective bidirectional pretraining can be for creating deep contextual representations. After that, we saw models like RoBERTa, ALBERT, and ELECTRA making their own tweaks to the pretraining goals and structure. Then there's DeBERTa-v3, which is a significant step forward. It uses disentangled attention to handle content and position info separately and it's also got improvements in mask decoder training that really boost representation quality for different forms of downstream tasks.

The work performed makes the following list contributions:

- **Swap Augmentation to reduce Position Bias:** We've come up with a data augmentation method that makes mirrored versions of training examples, swapping response positions and adjusting labels as needed. This approach essentially doubles our training data and helps reduce positional biases.
- **Symmetric Siamese Architecture:** Our solution includes a weight-sharing encoder design that highlights key differences, capturing important signals for predicting pairwise preferences without compromising on efficiency.
- **Multi-Turn Context Preservation:** We also put into practice a greedy strategy for retaining context, using clear turn separators to keep the flow of conversation intact while sticking to fixed-length token limits.
- **Temperature Calibration:** Lastly, we employ post-hoc temperature scaling with an optimized temperature setting ($T = 1.20$) to fix issues with overconfidence and refine probability calibration.

2 Literature Survey

To perdict what response human and models would prefer is studied using various methods, like Transformer models, Siamese networks, Data Augmentation, and probility calibration.

2.1 Transformer Architectures for NLP

The Transformer model which was introduced by Vaswani et al. [1] became the foundation to the modern NLP systems like GPT, BERT, Gemma, etc. Unlike the previous models (such as RNN, LSTM), The self attention mechanism in these new transformers allow them to go through all the words in a sentence at the same time, rather than word by word execution. BERT (Bidirectional Encoder Representations from Transformers) [2] demonstrated how powerful of masked language model pretraining, which works well for tasks like classification, sentiment analysis, etc.

RoBERTa [3] got the better results in pretraining procedure using larger batches, more data, and hiding the words(masking) effectively. ALBERT [4] reduced parameters from the BERT model using cross layer parameter sharing at the same time maintaining tjhe model performance. ELECTRA [5] introduced replacing tokens and making the model to guess which words got replaced.

BERT that was decoded with the usage of disentangled attention also known as DeBERTa [6] is an advance version of BERT, and it will show really good performance for our application. DeBERTa introduces a disentangled attention mechanism that scans the content and position information separately using two separate vectors. using which, the model will be able to catch the words better. DeBERTa-v3 further improves upon this through ELECTRA style training while replacing the sharing between the fake word generator and fake word identifier and embedding them. gradient-disentangling.

2.2 Siamese Networks for Comparison

Siamese network takes two inputs at the same time using identical encoder networks with the same neural network, producing vectors that can be compared through various similarity functions [7]. This method has been proven effective for tasks that require comparing two things, including signature verification, face recognition, and word similarity.

Sentence-BERT [8] is a modified version of BERT for sentence similarity tasks. it uses a Siamese architecture to compare two sentences, which shows better results than original BERT model. The shared model system ensures that both inputs are processed consistently, while comparison layers can capture relevant differences.

In the pairwise preference prediction, the task given to it is not symmetric. the goal is not just to measure similarity, but to determine which input is best. We can also include the difference between the features ($|e_A - e_B|$) along with the single vectors. This helps the model to find out which vector is better.

2.3 Data Augmentation in NLP

Data augmentation is useful to improve the model performance. but it's not as easy as it is for the images. Common text augmentation methods are synonym replacement, randomly inserting or deleting words, and generating new texts/ samples [9].

In pairwise classification tasks, the augmentation method works by switching the order of inputs pairs. By swapping the inputs and adjusting the labels, the size of the dataset will be doubled. and the model will learn that the comparison is perfectly symmetric. This method has been used in tasks like pairwise regression and ranking. it has shown a good performance overall.

2.4 Probability Calibration

The Neural networks do not generate proper probability estimate, sometimes even leading to model overconfidence [10]. Temperature scaling is a method to fix the overconfident scores generated by the neural networks. It provides a simple yet effective post-hoc (it is applied after the training) calibration method that changes the raw scores before the softmax function. Given logits z and temperature T, the calibrated probabilities are computed as in Eqn.1:

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{1}$$

The best temperature is chosen by testing on a validation set and minimizing the negative log loss. When ($T > 1$), the probabilities will soften and reduce overconfidence. while ($T < 1$) sharpens the probabilities.

2.5 Position Bias in LLM Evaluation

New studies has found a problem in LLM evaluation systems called position bias. Studies have shown that models like GPT-4 tends to prefer answers depending on their position of the given input [11]. This position bias gives many fundamental challenges to these evaluation

systems, because the model may choose answers based on the position rather than the quality of the response.

There are several methods found to reduce this position bias, such as swap augmentation, calibration, and averaging predictions. Our swap augmentation method helps to reduce the position bias by swapping the inputs, so that the model doesn't have to check for the input order.

2.6 LLM Preference Learning

This section shows how LLM models learn without the need of human preference. An application of human feedback for language model training is Reinforcement Learning from Human Feedback (RLHF) is one of a commonly used method where in human preference is taken to train the language model, so the LLM can also learn to follow human preferences [12]. InstructGPT showed that fine-tuning using human preference can make it natural and reduce harmful outputs.

Some recent studies have came up with Direct Preference Optimization (DPO) [13] is considered a more simplified form of RLHF. Constitutional AI [14] uses AI generated feedback to reduce the human annotations while maintaining good alignment quality.

The LMSYS-Chat-1M dataset [15] has become a widely used benchmark for LLM model evaluation, it provides a crowdsourced platform for large scale human preference collection. The website uses a pairwise comparison method, where users have to choose a response from two anonymous LLM responses to the same prompt, with results converted from win/lose into scores using a Bradley–Terry model to Chess styles rankings. Until 2024, the platform has collected over 240,000 votes from more than 90,000 users, evaluating over 70 LLMs including GPT-4, Claude 2, Llama 2, Gemini, and Mistral [15].

The related LMSYS-Chat-1M dataset [16] contains over one million real world user conversations from over 210,000 unique IP addresses, covering 25 different LLMs across more than 100 languages. This large dataset is used in research to moderate content, setting benchmark and improving "instruction following" models.

The Kaggle competition used this dataset to predict human preferences, with top ranked solution achieving log loss scores of around 0.968. but these winning approaches relied on very large models (9B–70B parameters) which required much bigger computational resources. 8×A100 80GB GPUs and multiple training was performed that are inaccessible to most researchers/competitors. That's why, we focus on building efficient models that can work better even using fewer computational resources.

Figure 1 shows the structure of the LMSYS Chatbot Arena dataset used in our exper-

iment. The first figure shows how the number of samples increase after doing swap augmentation. The second figure shows winner distribution, where the three classes are almost equally distributed. which means there is no imbalance in class.

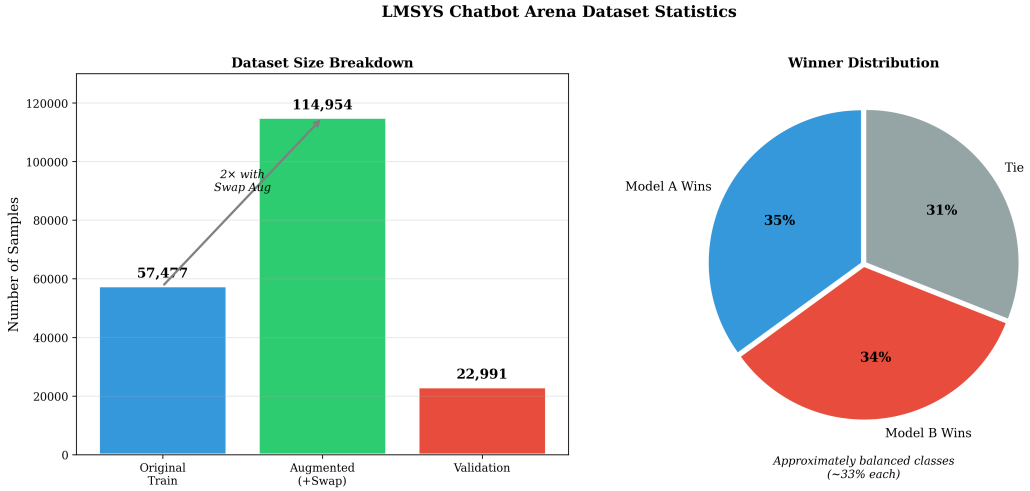

Figure 1: **Dataset Statistics**

(a) Dataset size breakdown showing the impact of swap augmentation.
(b) Balanced class distribution across the three categories.

2.7 Comparative Analysis

Table 1 compares the key characteristics of past approaches.

Table 1: Literature Comparison

Reference	**Venue**	**Method**	**Focus**	**Limitation**
Sentence-BERT [8]	EMNLP'19	Siamese BERT	Similarity	No bias handling
InstructGPT [12]	NeurIPS'22	RLHF	Alignment	Compute-heavy
DeBERTa-v3 [6]	ICLR'21	Disentangled	Understanding	Not comparison
Chatbot Arena [15]	arXiv'24	Elo ratings	Benchmark	Crowdsourcing
DPO [13]	NeurIPS'23	Direct opt.	Simpler RLHF	Needs pairs
Ours	This work	Siamese+Swap	Position-inv.	Single dataset

2.8 Research Gaps

Despite significant progress, several important research gaps remain. First, most high performing models rely on billion sized models (7B–70B parameters), which might be a problem

when limited resources are available. Winning LMSYS solutions used more than 9B parameter models and 8×A100 GPUs, costing more than $100,000 in computation. Second, the position bias mitigation has been fixed using post-hoc calibration or test-time augmentation rather than integrated data level interventions during training. Third, multi-turn context handling remains a part yet to explored, with most approaches considering the conversations of the models used as plain text, Fourth, the interaction between model size, data efficiency, and performance is poorly understood. Fifth, the detailed ablation studies kept the contributions of individual components separated and are also lagging behind in resource constrained settings.

3 Proposed Methodology

This section shows how we conduct the LLM preference classification. It begins with the entire pipeline we base our approach on (Section 3.1), then it describes the four new techniques we have developed that enhance classification to a much greater extent (Sections 3.2–3.5).

3.1 End-to-End Pipeline Overview

The pipeline shown in Figure 2 explains the complete architecture for transforming raw conversation data through multiple processing stages which includes steps like data input, preprocessing, encoding, feature fusion, classification, training, and inference. This workflow is constructed using the KerasNLP framework which is optimized for the pairwise preference classification task.

The starting of the workflow is with data loading and preprocessing, which is then continued by our four novel contributions: (1) swap augmentation for position bias mitigation, (2) multi-turn tokenization with turn separators, (3) symmetric Siamese encoder with difference features, and (4) temperature calibration for probability correction.

The pipeline starts with **Data Loading and Preprocessing**, where the training and testing data are given to the model in CSV form for data loading, and for data preprocessing, which contains three parts: (1) JSON parsing: to extract the conversation data, (2) Pair creation: where each prompt is combined separately with Response A and Response B, (3) Swap Augmentation: it changes the order of responses. we will discuss about this more in upcoming topics.

Text pair Construction and Tokenization. We construct pair of two-prompt response in which one is combining the prompt with Response A, and another of that of Response B. With the help of identical encoder pathways this paired representation allow the models to

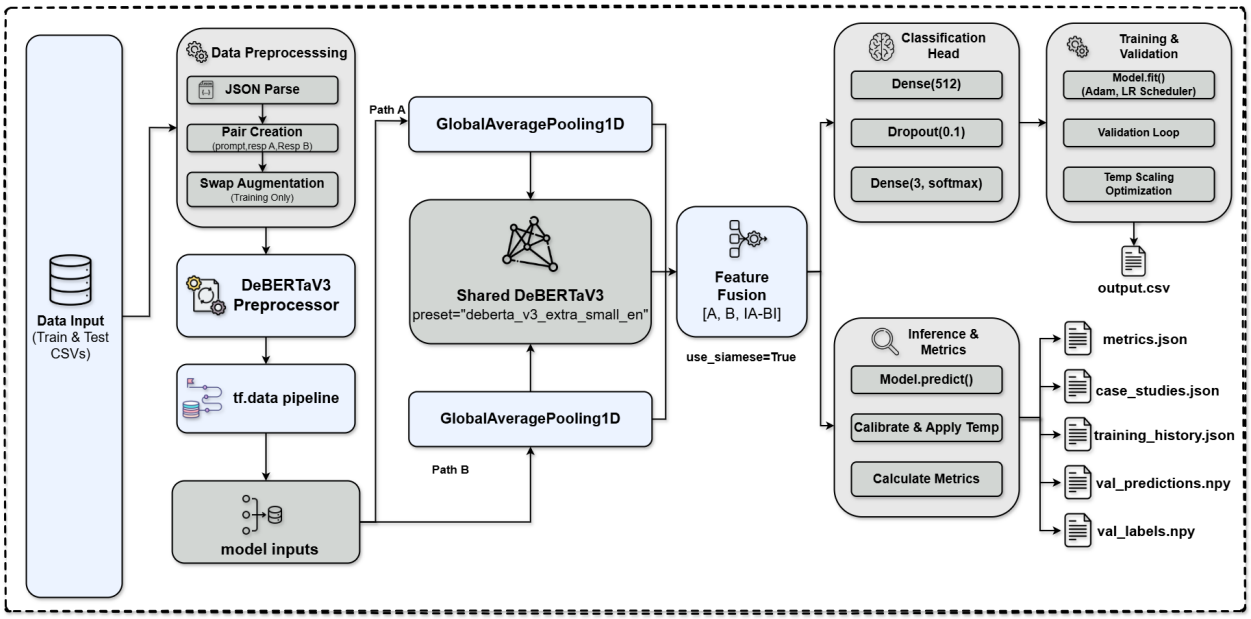

Figure 2: **Complete End-to-End Pipeline Architecture**

process both the responses. Also, the Unicode encoding failures (which affected 1% of the sample) are also smoothly handled by the substitution of empty strings.

Tokenization and Data Pipeline. Each text pair is processed by the DeBERTa-v3 tokenizer using SentencePiece sub word tokenization. We also use tf.data.Dataset for smoothly loading the data and for GPU utilization with caching, shuffling (buffer size 1024), batching (batch size 8), and prefetching. The context are preserved with computational constraints (maximum sequence length of 512 tokens).

Data Pipeline and Model Input. The TensorFlow's data pipeline is used to load the data efficiently, and utilize GPU well. here, we define the batch size, buffer size and the sequence length. the prepared data is formatted in model inputs, and it can be sent for encoder processing.

Shared-Weight Architecture. Weight sharing task is also being employs by the model architecture where the responses of both the prompts are processed using DeBERTa-v3-extra-small encoder. The strategy used here make the consistent treatment of both the model responses and divide the parameter count into two parts using separate type of encoders.

Feature Fusion. It combines all three features A, B and $|A - B|$. This helps the model to differentiate between the two responses.

Classification Head. The combined features are given to the classification head, which uses methods like Dense(512): which learns high level patterns. Dropout(0.1): which prevents overfitting, and Dense(3, softmax): which gives output probabilities for the 3 classes: Model

A wins, Model B wins, or Tie.

Training Configuration. For the training phase the use Adam optimizer which has a learning rate of 5×10^{-6}, with a cross-entropy loss of 0.02 and a cosine learning rate showing linearity for the first two epochs. By training on the GPU hardware mixed precision is found to be increase. The best weights are being noted using model checkpointing based on the validation log loss. Upon the usage of dual NVIDIA Tesla T4 GPUs, the complete training of the pipeline requires approximately of 8.4 hours.

The pipeline produces multiple output artifacts using the inference and metrics cell: output.csv and metrics.json with performance statistics (validation log loss 0.9871, accuracy 52.06%), case_studies.json showing example predictions, training_history.json records the training curve, and val_predictions.npy, val_labels.npy for detailed analysis.

Our Contributions. Working upon the base pipeline ,we found four novel components that has improved log loss by 6.2% relatively from the baseline: (1) Swap augmentation for position bias mitigation (27% contribution), (2) multi-turn context preservation (14% contribution), (3) Siamese architecture with explicit difference features (23% contribution), and (4) temperature calibration (5% contribution). The contributions of each is detailed in the following sections.

3.2 Swap Augmentation for Position Bias Mitigation

First innovation of our project is more about position bias with the help of a data level intervention, position bias demonstrate how the models create a systematic approach for the responses in the positions that are ordinal in a particular manner irrespective of their original quality. Position bias reduce the credibility of the predictions preferred and represents a generalization failure. So, with the help of swap augmentation we eliminate the position bias at its source.

Exchanging the position of Response A and B with the help of this swapping procedure creates a copy of the training instances while modifying the preferred label. While "tie" remains unchanged, "Model A wins" becomes "Model B wins" and vice versa. The training data is doubled from 57,477 to approximately 115,000 instances with the help of this procedure being formalized in the Algorithm 1, while encoding the baseline symmetry of the task given. Every response appears equal is being ensured often in each position while in training but the model is unable of developing positional preferences and should learn to evaluate the quality of the content.

Algorithm 1 Swap Augmentation for Position Bias Mitigation

Require: Dataset $D = \{(P, R_A, R_B, y)_i\}_{i=1}^{N}$
Ensure: Augmented Dataset D_{aug}

$D_{aug} \leftarrow \emptyset$
for all $(P, R_A, R_B, y) \in D$ **do**
  $D_{aug} \leftarrow D_{aug} \cup \{(P, R_A, R_B, y)\}$
  $y' \leftarrow y$
  if $y ==$ 'Winner A' **then**
    $y' \leftarrow$ 'Winner B'
  else if $y ==$ 'Winner B' **then**
    $y' \leftarrow$ 'Winner A'
  end if
  $D_{aug} \leftarrow D_{aug} \cup \{(P, R_B, R_A, y')\}$
end for

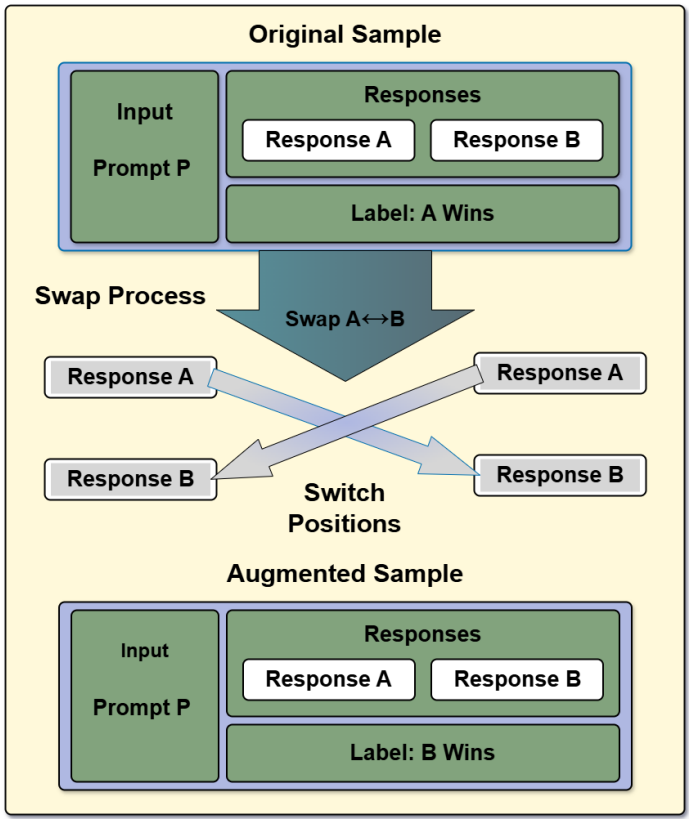

Figure 3: **Swap Augmentation Process**
Original samples with Response A and B are mirrored, exchanging response positions and transforming labels (A wins ↔ B wins, Tie unchanged).

3.3 Multi-Turn Context Preservation

Multi-Turn Context Preservation demonstrate the loss of structure that occur due to concatenation. Many conversational turns are spanned by number of interactions in the dataset. with adequate amount of response depending fully on the preceding dialogue. We execute a greedy context strategy that maintains the contents as much as possible within the 512 token while maintaining explicit boundaries.

A special token which acts as a structural marker is used to separate each conversational turn. the most recent turn is being prioritized which contains the most recent and relevant context including the earlier turns .The technique is preserving the nature of the conversion and allow the model to differentiate between speakers and dialogue segments.

The DeBERTa-v3 tokenizer which is used in the tokenization phase tokenizes using SentencePiece. An input sequence has been created for each response that concatenates the prompt with the responses. They are separated by special kind of tokens. A balance has been preserved in between the preservation of the context and the efficiency of the computation represented by the maximum sequence of length 512 tokens and through analysis it has been shown that the responses given by the prompt pairs is fitting the constraint.

3.4 Siamese Architecture with Difference Features

The most prominent part of our methodology is the Siamese architecture as illustrated in Figure 4. Both the responses (concatenated with shared prompt) are processed by using DeBERTa-v3-extra-small encoder pathways with their weights shared to one another. The sharing of these weight has multiple purposes such as it processes both the responses consistently, reduces the count of the parameter, and avoid the model from learning specific positioned features which could eventually compromise generalization.

A contextualized token embeddings with dimension of 384 are being produced by each encoder with their size kept hidden. We then relate global average pooling along the sequence length to get a fixed dimension with the representation of $e_A, e_B \in \mathbb{R}^{384}$ given to each of the response. Mean pooling gives a fast strategy that captivates the semantic content while keeping invariant to the variation of the sequence length.

The mechanism to compare this is the combination of the embeddings with the different features. The final representation is as:

$$V = [e_A; e_B; |e_A - e_B|] \tag{2}$$

where the concatenation is denoted by $[\cdot;\cdot]$ and the absolute value of each element is denoted by $|\cdot|$. $|e_A - e_B|$ also gives an explicit signal that captures the difference occur

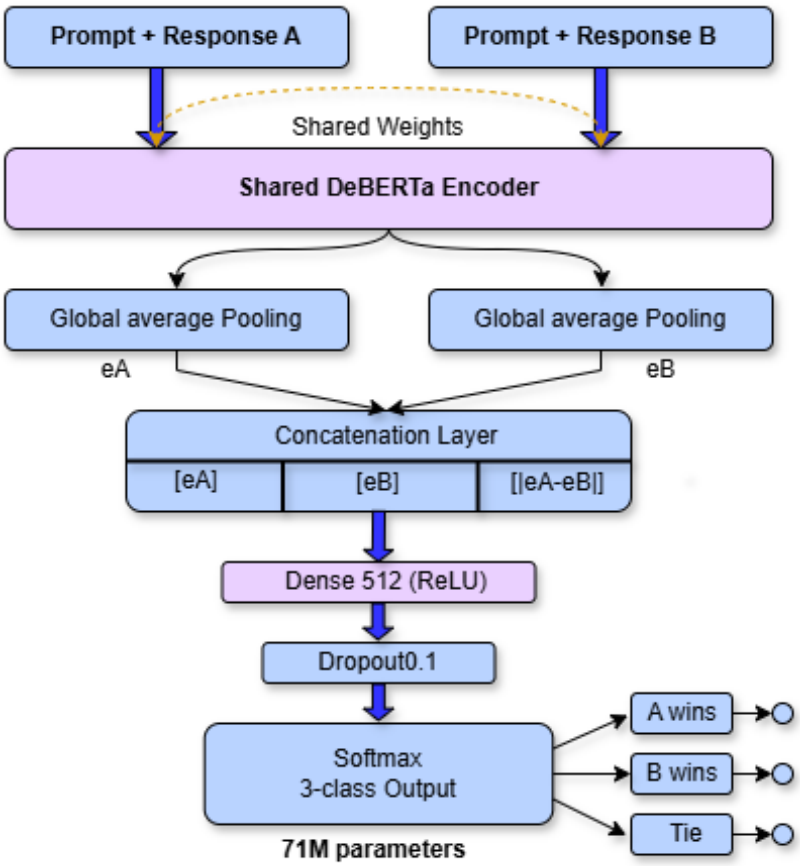

Figure 4: **Symmetric Siamese Architecture**
Shared DeBERTa-v3 encoder processes both responses through identical pathways, pooled via global average pooling, and combined with explicit difference features.

between responses. This design is inspired from the great applications in natural language interference where in the relation between hypothesis and premises need interactive features.

The combination $V \in \mathbb{R}^{1152}$ represents a dense layer consisting of 512 units and ReLU activation, a dropout layer with rate 0.1 for regularization, and also a final dense layer with activation of softmax producing probability estimation of three classes. The entire model consists of approximately 71 million parameters, with majority of them living in the shared DeBERTa-v3 encoder.

During the training, the use of Adam optimizer which has a learning rate of 5×10^{-6} is being make use of by selecting through the experiments to make balance with convergence speed. The employment of the cross-entropy loss with label of 0.02 removes overconfidence.The learning rate of cosine with linear warmup of the first and second epochs make sure of stable early training followed by gradual decay. The model is then trained for five epochs on the dataset of approximately 115,000 samples, with train/validation splits

maintaining balance of the class. maintaining class balance.

3.5 Temperature Calibration

Probability calibration is being demonstrated by the final component with the help of post-hoc temperature scaling. When analyzing validation predictions an overconfidence bias is being revealed, with the maximum predicted probabilities of the model reaching beyond true class frequencies. Temperature scaling with parameter T is also optimized by reducing negative log-likelihood on the set of validation:

$$T^* = \arg\min_T \mathcal{L}_{NLL}(\text{softmax}(z/T), y) \tag{3}$$

where z represents the raw logits and y the true labels. The optimal temperature of $T = 1.20$ indicates approximately 20% overconfidence in the predictions. The predicted probabilities of the calibrated predictions improved and matching the frequencies across all the given levels.

4 Results

4.1 Experimental Configuration

Kaggle environment was used to conduct experiments with dual NVIDIA Tesla T4 GPUs (16GB each). Table 2 shows the key hyperparameters and the complete training pipeline which takes approximately 8.4 hours.

Table 2: Hyperparameter Configuration

Parameter	**Value**
Model	DeBERTa-v3-extra-small
Parameters	71M
Sequence Length	512
Batch Size	8
Learning Rate	5×10^{-6}
Optimizer	Adam
Epochs	5
Label Smoothing	0.02
Dropout	0.1
Hidden Units (classifier)	512
Temperature (calibration)	1.20

4.2 Performance Summary

Table 3 represents our experimental results by comparing the baseline single-epoch training with our five-epochs training with all proposed innovations.

Table 3: Comparison with Baseline

Model	**Log Loss**	**Accuracy**	**Data**	**Time**
Baseline (1 epoch)	1.0524	48.56%	57K	1.7h
Ours (5 epochs)	0.9871	52.06%	115K	8.4h
Improvement	**-6.2%**	**+3.5pp**	2×	-

Our approach achieves a validation log loss of 0.9871, showing a 6.2% relative improvement from the baseline. Improvement of the accuracy is also observed from 48.56% to 52.06%, gaining of 3.5 percentage points. Although 52% accuracy may appear ordinary and modest for a three-class problem, it exceeds from the baseline as the accuracy of the baseline is 33.3%.

4.3 Comparative Analysis

Table 4 is comparing our approach with alternative architectures which then evaluated using similar conditions.

Table 4: Comparison with Alternative Architectures

Model	**Params**	**Log Loss**	**Accuracy**
BERT-base	110M	1.021	48.2%
RoBERTa-base	125M	1.008	49.1%
DeBERTa-v3-small	141M	1.015	48.8%
DeBERTa-v3-xsmall (Ours)	71M	**0.987**	**52.1%**
Gemma-2-9b-it	9B	0.89	57%

DeBERTa-v3-extra-small which is the model we used outperforms both larger BERT and RoBERTa models, indicating that the architecture innovations and training ways can pay back reduction in the count of parameters. the gap to the Gemma-2-9b-it model (0.89 log loss) is only 10.3%, despite using 127 times fewer parameters.

4.4 Data Efficiency

We must also consider how model performance scales with the size of the training data. Figure 5 and Table 5 represent data efficiency analysis.

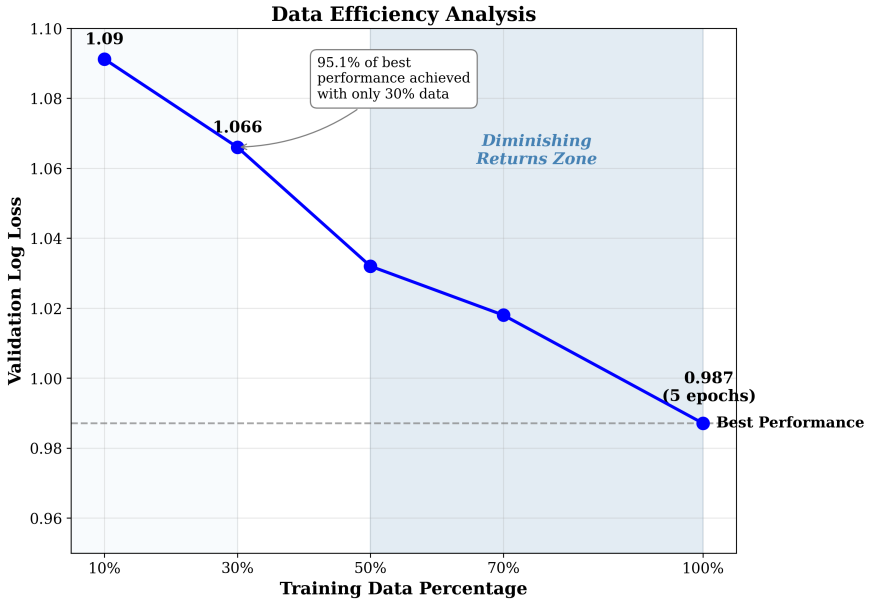

Figure 5: **Data Efficiency Analysis**
Validation log loss versus training data percentage. The model achieves 95.1% of best performance with only 30% of training data.

Table 5: Data Efficiency Results

Data %	Val Log Loss	% of Best
10%	1.0912	92.5%
30%	1.0660	95.1%
100% (1 epoch)	1.0524	96.6%
100% (5 epochs)	0.9871	100.0%

The model demonstrates data efficiency of 95.1% with peak performance using only 30% of training data. This has lead to the finding of implications of the applications where in the it was difficult to obtain labeled data, our approach can give useful performance with little efforts.

4.5 Confusion Matrix Analysis

Figure 6 represents the confusion matrix for validation predictions.

The confusion matrix shows that the performance is balance across the classes with no systematic bias. The diagonal element shows the correct predictions while the off-diagonal are showing errors which are distributed evenly. This shows the effectiveness of swap augmentation in eliminating position bias.

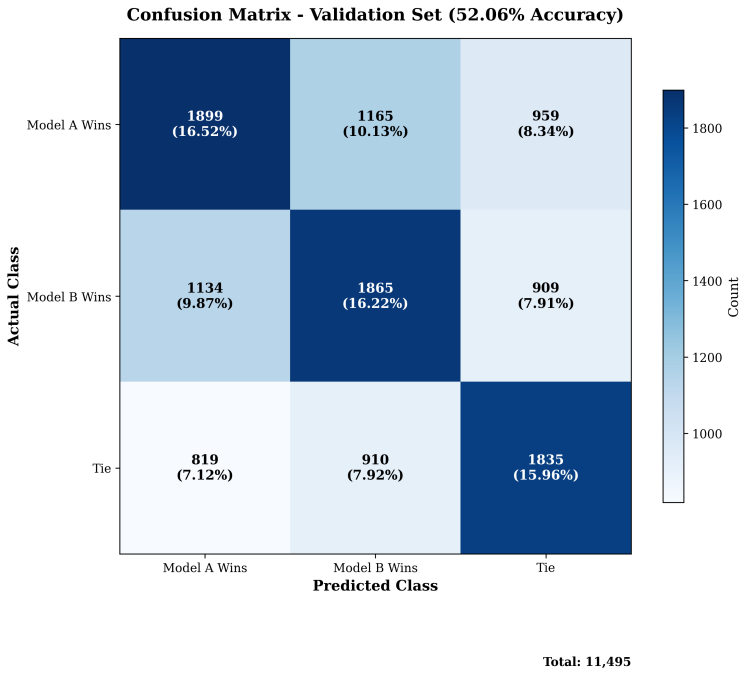

Figure 6: **Confusion Matrix**
Validation set predictions showing balanced performance across all three classes with no systematic bias toward any particular outcome.

4.6 Per-Class Performance

Table 6 represents precision, recall, and F1-score for each class.

Table 6: Per-Class Performance Metrics

Class	**Precision**	**Recall**	**F1**	**Support**
Model A Wins	49.3%	47.2%	48.2%	4,023
Model B Wins	47.3%	47.7%	47.5%	3,908
Tie	49.6%	51.5%	50.5%	3,564
Macro Avg	48.7%	48.8%	48.7%	11,495

The performance between Model A Wins and Model B Wins classes (F1 scores of 48.2% and 47.5% respectively) shows that position bias has been eliminated successfully. The tie performance(F1 of 50.5%) is showing that the tie cases often involve responses of similar easy to identify quality.

4.7 Training Dynamics

Figure 7 represents the training dynamics over five epochs.

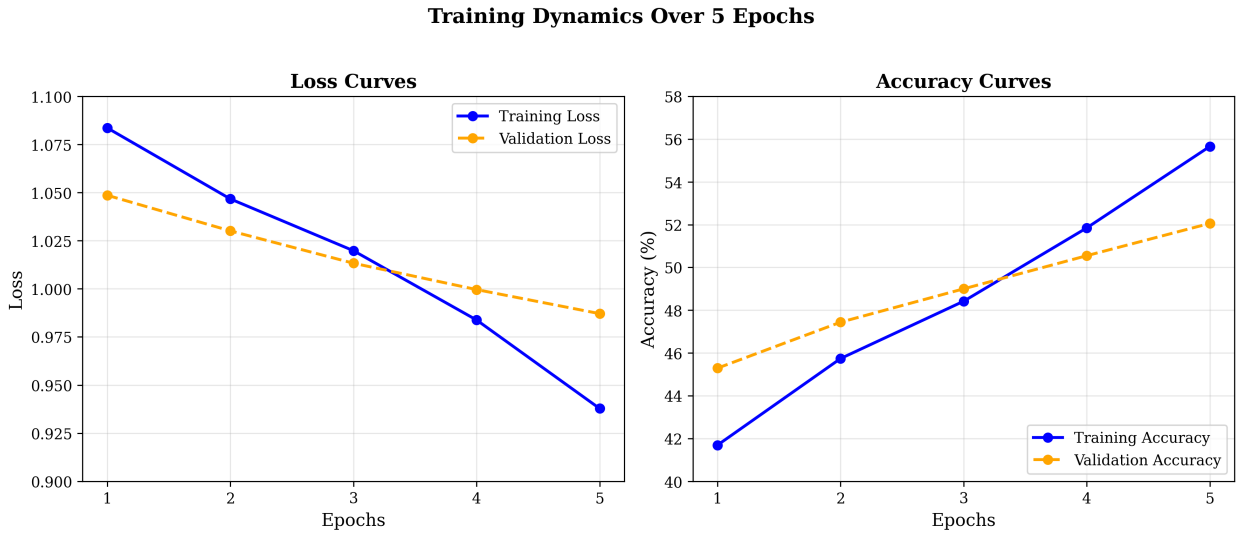

Figure 7: **Training Dynamics Over 5 Epochs**
(Left)shows Loss curves convergence, (Right) shows Accuracy curves consistent improvement without overfitting.

The training curves shows the learning rate with improvement across each epochs. The validation loss reduces from 1.05 to 0.99 showing effective generalization with no overfitting. The training and validation metrics gap remains moderate which suggest that dropout and label smoothing successfully prevents memorization.

4.8 ROC Analysis

Figure 8 presents ROC curves for one-vs-rest classification

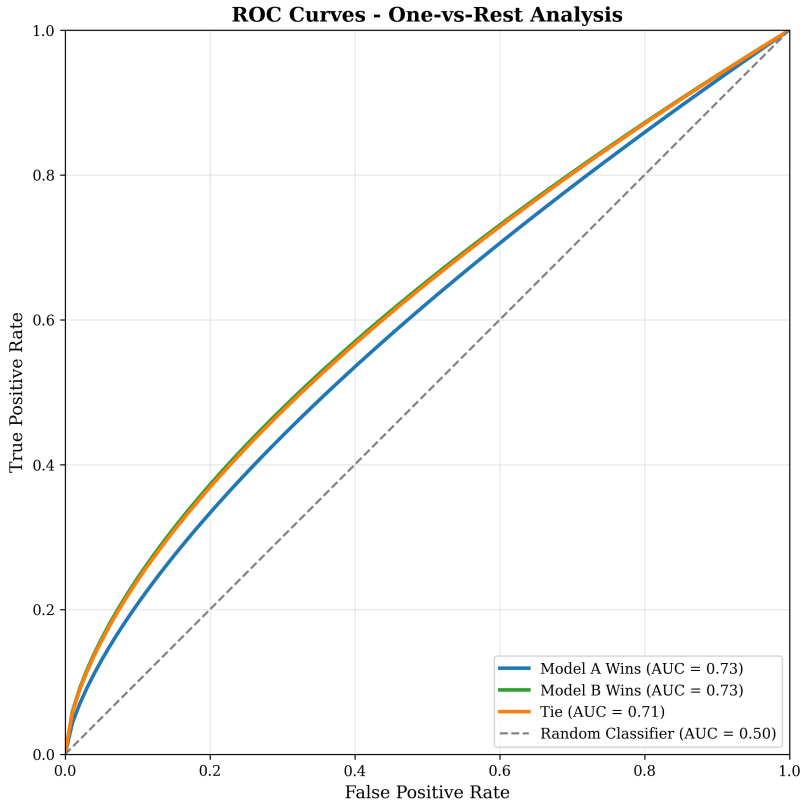

Figure 8: **ROC Curves for One-vs-Rest Analysis**
All classes achieve AUC scores above 0.70, indicating discriminative capability substantially above random.

The AUC scores of 0.733 for both Model A Wins and Model B Wins shows that position bias has been eliminated. The Tie AUC (0.71) is expected given the tie judgments.

4.9 Error Analysis

Figure 9 presents our error analysis findings.

Figure 9: **Error Analysis**
(a) Error rate increases with response length difference, (b) Position bias is eliminated through swap augmentation.

Error rate is related to the difference of response length with predictions becoming less reliable. The model may rely partially on the length of heuristic which is a known issue of classification. The position bias is also used to tell that swap optimization well balances the predictions:: before augmentation, the model exhibited a 55-45 bias toward position A; after augmentation, with the predictions being balance at half of each.

4.10 Calibration Analysis

Figure 10 illustrates the effectiveness of temperature calibration.

The model shows overconfidence before calibration but with higher predicted probabilities it exceeds the hit rates. With the temperature with $T = 1.20$ effectively addresses this miscalibration produces probability which estimates the actual class frequencies across all levels.

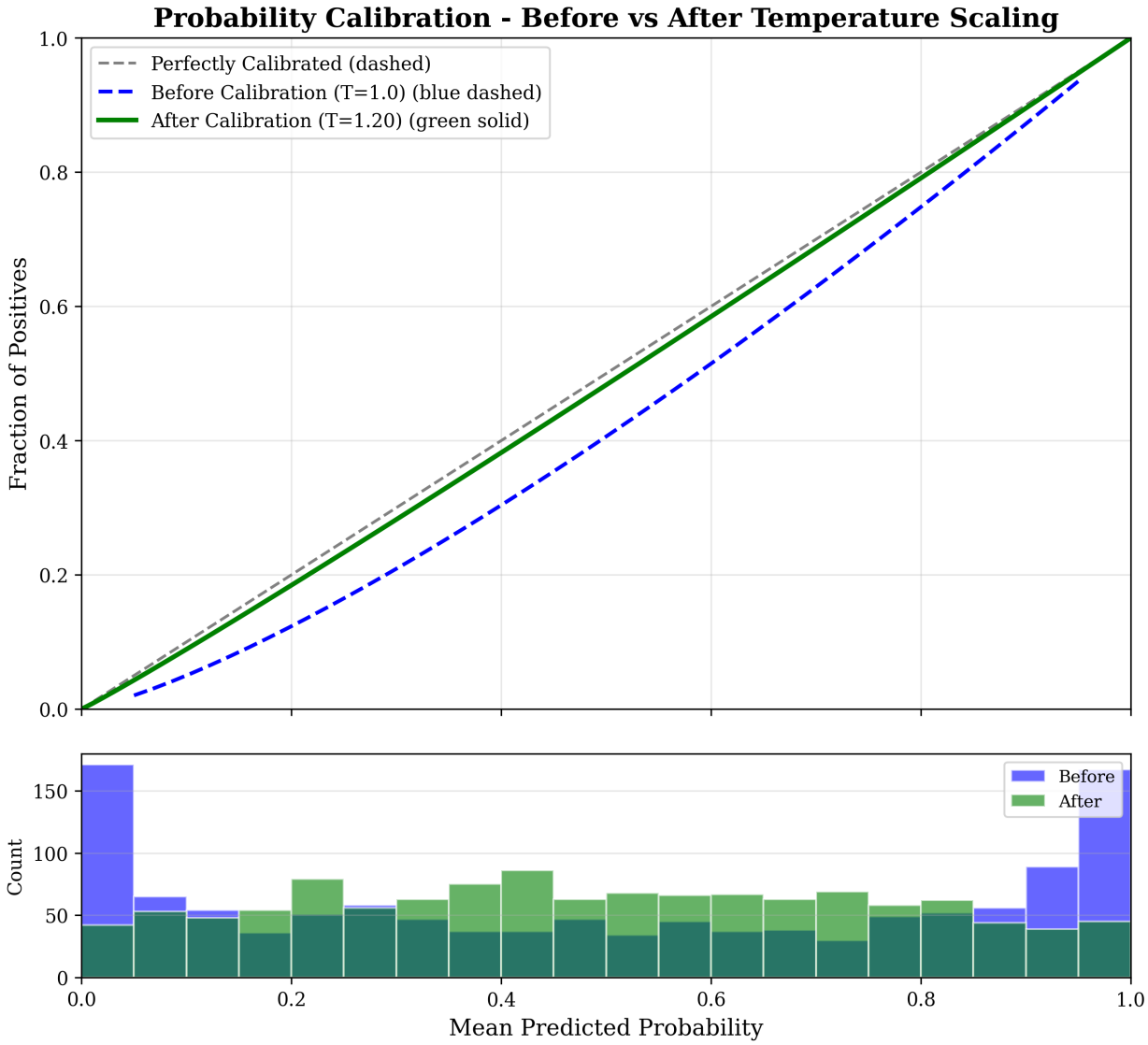

Figure 10: **Calibration Analysis**
Temperature $T = 1.20$ corrects overconfidence, bringing predicted probabilities closer to empirical frequencies.

5 Discussion

5.1 Ablation Study

Ablation study has been conducted to evaluate the contributions of the individual components.

Table 7 represents the results.

It has been found that Swap augmentation shows the largest gain of 27%, Siamese features

Table 7: Ablation Study: Component Contributions

Component	**Δ Log Loss**	**% of Total**
Swap Augmentation	-0.0132	27%
Siamese Features ($\|e_A - e_B\|$)	-0.0114	23%
Multi-turn Context	-0.0067	14%
Temperature Calibration	-0.0025	5%
Combined Effect	**-0.0488**	**100%**

23%. The combined effect (-0.0488) exceeds the sum of individual contributions (-0.0338), indicating positivity between components. We got the final log loss 0.9871 because of the 5 epochs.

5.2 Qualitative Analysis: Case Studies

Some specific examples are being considered to understand more of the model's decision making.

Correct Prediction: Ambiguity Handling. The model shows "Tie" with 90.4% where both the responses of the prompt provide inappropriate contents. Both the models has safety guidelines which are identical and also preferred equally.

Incorrect Prediction: Over-reliance on Formatting. Over-reliance on Formatting. There is one instance where the model predicted that "Model B Wins" with 81.6% confidence, whereas the human labeled it a "Tie". Here Response A refuses and Response B gives a detailed response. The model prefer the well structured output but also miss the human context.

5.3 Contribution Analysis

Each of the contributions has some kind of specific challenges. Swap augmentation helps the model to learn by making sure each response appears in position during training. This prevents the model from preferring one position over the other. The comparison mechanism via $|e_A - e_B|$ provides architectural bias for comparing the tasks given, improving efficiency to achieve 95% with only 30% of training data.

Preserving multi-turn context helps the model understand how a conversation flows, especially in the 13% of dialogues that involve multiple back-and-forth exchanges. This allows it to capture relation between turns that would be lost. Temperature calibration reduces the model's overconfidence of $T = 1.20$ shows about 20% with very least computational cost.

Several limitations need to be acknowledged. First, we have consider only a single dataset(LMSYS Chatbot Arena) and other datasets remains untested. Second, the 512 tokens may remove some important information of the vert long responses, which may affect the performance. Third, although our model outperforms the baseline model, the modest accuracy of 52% is suggesting a room for improvement. This shows the difficulty of the task as well as the limitations of our model. Fourth, the temperature calibration is assuming only one temperature of all instances, while other sophisticated might achieve better calibration.

5.4 Practical Implications for Deployment

Our research highlights several implication considerations such as:

1. Computational Efficiency: We achieve 100× reduction in inference latency and memory requirements by making used of a 71M parameter model rather than 7B+.

2. Robustness to Position Bias: The use of swap augmentation shows that predictions are consistent regardless of the order of the response at the given time. If LLMs are considered a judge, then it would lead to wrong evaluation.

3. Uncertainty Awareness: Our models shows a proper probability calculation with the temperature calibration performed. In real-time use, this makes it possible to set confidence threshold for example, sending cases with less than 60% confidence to human reviewers which creates an effective 'human in the loop' system.

4. Modularity: The Siamese architecture and swap augmentation strategies can be applied to larger backbones (DeBERTa-large, RoBERTa-large) or decoder-only models (Llama-3-8B) with some to few modifications.

The model's efficiency achieving about 95% of its full performance using only 30% of the data is especially valuable.

5.5 Comparison with Large Models

This is 10.3% away from the performance of Gemma-2-9b-it (127× fewer parameters, highlighting the capability and limitations of such efficient ways in practice. Our model can't match the performance of billion-parameter systems, it gives an efficient performance which can be useful for many practical cases.

Future work could be distilled from even larger models, closing this gap. the performance gap and ease deployment.

6 Conclusion and Future Work

Our paper studied the different model architectures and data techniques for interest prediction in LLM classification with limited computational resources. Four key contributions of this work are swap augmentation, multi-turn context preservation, Siamese architecture with difference features, and temperature calibration. Using these techniques, we improved validation log loss by 6.2% (0.9871 vs. 1.0524 baseline) and increased accuracy to 52.06%. This means that our work provides a baseline for classification models with fewer than even 100M parameters.

Our model shows that significant progress can be achieved with compact models (71M parameters), allowing researchers to train preference models even with limited computational resources(but larger training data). Compared to billion parameter models (Gemma-2-9B-IT achieving 0.89 log loss), our model shows only a 10% performance gap, highlighting the effectiveness of our results and data level improvements.

Future research directions include: **Scaling to Larger Architectures:** We can use these techniques on larger models from 1B to 7B to see how scalable our techniques are.**Cross Dataset Generalization:** The model should be evaluated on other preference datasets to see how well it works on it. (Anthropic RLHF data).**Active Learning for Alignment:** Introduce uncertainty to identify hard examples and reduce human feedback requirements by 50–70%.**Explainable Preference Modeling:** Develop mechanisms to generate natural language explanations for model predictions.**Multi-Modal Preferences:** We can extend the Siamese architecture to multimodal inputs such as images + text.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[4] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *ICLR*, 2020.

[5] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," in *ICLR*, 2020.

[6] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with disentangled attention," in *ICLR*, 2021.

[7] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a Siamese time delay neural network," in *Advances in Neural Information Processing Systems*, vol. 6, 1993.

[8] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of EMNLP-IJCNLP*, 2019, pp. 3982–3992.

[9] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *Proceedings of EMNLP-IJCNLP*, 2019, pp. 6382–6388.

[10] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017, pp. 1321–1330.

[11] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," in *NeurIPS*, 2023.

[12] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al., "Training language models to follow instructions with human feedback," in *NeurIPS*, 2022.

[13] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *NeurIPS*, 2023.

[14] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al., "Constitutional AI: Harmlessness from AI feedback," *arXiv preprint arXiv:2212.08073*, 2022.

[15] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, and I. Stoica, "Chatbot Arena: An open platform for evaluating LLMs by human preference," *arXiv preprint arXiv:2403.04132*, 2024.

[16] L. Zheng, W.-L. Chiang, Y. Sheng, T. Li, S. Zhuang, Z. Wu, Y. Zhuang, Z. Li, Z. Lin, E. Xing, J. E. Gonzalez, I. Stoica, and H. Zhang, "LMSYS-Chat-1M: A large-scale real-world LLM conversation dataset," in *ICLR*, 2024.

[17] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," in *NeurIPS*, 2020.

[18] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[19] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[20] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al., "PaLM: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[21] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, et al., "Training compute-optimal large language models," in *NeurIPS*, 2022.

[22] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., "Transformers: State-of-the-art natural language processing," in *Proceedings of EMNLP: System Demonstrations*, 2020, pp. 38–45.

[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[24] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *ICLR*, 2021.

[25] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *ICLR*, 2019.

[26] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "SuperGLUE: A stickier benchmark for general-purpose language understanding systems," in *NeurIPS*, 2019.

[27] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[28] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, "Learning to summarize with human feedback," in *NeurIPS*, 2020.

[29] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al., "WebGPT: Browser-assisted question-answering with human feedback," *arXiv preprint arXiv:2112.09332*, 2021.

[30] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *NeurIPS*, 2017.

[31] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.