CHAI for LLMs: Improving Code-Mixed Translation in Large Language Models through Reinforcement Learning with AI Feedback

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities across various NLP tasks but struggle with code-mixed (or code-switched) language understanding. For example, prior work benchmarking the performance of multilingual LLMs on code-mixed translation tasks has demonstrated that current state-of-the-art multilingual LLMs are ineffective in dealing with code-mixed languages. However, the question of how to improve the capability of multilingual LLMs to handle code-mixed language has not received any attention to date. In this paper, we tackle this research gap by proposing CHAI, a novel general-purpose framework for improving the ability of multilingual LLMs to handle code-mixed languages. CHAI relies on three novel contributions made in this paper. First, we explore the ability of LLMs to provide accurate annotations for code-mixed translation tasks. Second, we leverage this ability of LLMs as annotators to generate preference data for code-mixed translation tasks at scale, which are then used within a reinforcement learning from AI feedback (RLAIF) procedure to improve LLMs' capability on code-mixed tasks. Third, we conduct a rigorous experimental evaluation across various real-world datasets and settings. Our analysis shows that CHAI-powered LLMs outperform state-of-the-art open-source LLMs by 25.66% (in terms of win rate adjudicated by human annotators) in code-mixed translation tasks. This work represents a first step towards developing more inclusive code-mixed LLMs.

1 Introduction

Large language models (LLMs) have excelled at comprehending, producing, and interacting with human language across a wide variety of real-world use cases, e.g., drafting code in information technology (Tian et al., 2023), generating hypotheses in biology (Park et al., 2024), formulating therapeutic dialogue in mental health settings (Cheng et al., 2023), etc. LLMs have also seen widespread user adoption, e.g., ChatGPT reached 100 million users in two months after its launch, the fastest growth of any consumer application in history (Hu, 2023).

Unfortunately, the vast linguistic diversity across the globe still poses significant challenges for such emerging LLM-based technologies. In particular, recent studies (Zhang et al., 2023a; Gupta et al., 2024a) have shown that the ability of current LLMs to understand and generate language is heavily skewed towards monolingual English language queries, with a significant performance degradation reported in prior work (Gupta et al., 2024b) on tasks involving code-mixed language[1]. These results are highly problematic because they leave a large proportion of the global population — those using code-mixed language as their primary means of communication (which includes more than 1 billion people in India alone) — at a comparative disadvantage (Ramzan et al., 2021). To ensure that the benefits of LLMs can extend to these populations, it is crucial that the next generation of LLMs can understand, reason, and respond to/in code-mixed language.

In large part, this performance degradation on code-mixed tasks occurs because most current-day LLMs have been trained on large corpora of monolingual and/or multilingual text, with comparatively little explicit code-mixed corpora included during the pre-training phase of LLM training. This lack of inclusion of code-mixed corpora can be attributed to a (relative) lack of availability of large-scale code-mixed datasets on the Internet (Maguerresse et al., 2020). Despite this, prior attempts at augmenting LLMs to handle code-mixed language have mainly focused on injecting additional code-mixed text during the pre-training stage (Zhang et al., 2023c). At the same time, while some stud-

[1]Code-mixing, the fluid alternation between languages within a conversation or text, is a common linguistic phenomenon, especially in multilingual societies (e.g., India).

ies highlight the cross-lingual transfer ability of LLMs, these results do not effectively extend to code-mixed language, where inconsistencies in grammar, syntax, and context-switching further hinder model performance.

These challenges motivate us to explore - *Can we develop a general-purpose approach to improve the capability of LLMs in dealing with code-mixed tasks?* To tackle this main research question, we propose CHAI (**C**ode Mixed Understanding via **H**ybrid **AI** **I**nstruction), a novel general-purpose framework for improving the ability of multilingual LLMs to handle code-mixed language. CHAI relies on three novel contributions. First, we explore the ability of LLMs in providing accurate annotations for code-mixed translation tasks. We compare LLM annotation results with human annotations, and our results show that LLM labeled preferences (for code-mixed text) are highly correlated with human annotator preferences. Second, we leverage this ability of LLMs (to serve as a proxy annotator) to generate preference data for code-mixed translation tasks at scale, which is then used to develop a new code-mixed LLM through model alignment. In particular, we adopt a reinforcement learning from AI feedback (RLAIF) procedure to improve the capability of current-day LLMs to handle code-mixed language. To the best of our knowledge, we are the first to utilize model alignment for the code-mixing scenario. Third, we conduct a rigorous experimental evaluation across various real-world datasets and settings. Our analysis shows that LLMs powered with CHAI outperform conventional state-of-the-art LLMs by 25.66% (in terms of win rate adjudicated by human annotators) on code-mixed translation tasks. This work takes a first step towards developing more inclusive code-mixed LLMs, which can empower people from diverse linguistic communities.

2 Related Work

We discuss three primary areas of related work in this section.

LLMs on Code-Mixed Tasks. Zhang et al. (2023b) investigates LLMs' potential in the context of code-mixed tasks. They benchmark multilingual LLMs' performance across sentiment analysis, machine translation, summarization, and word-level language identification tasks. They argue that current multilingual capabilities in LLMs do not imply proficiency with code-mixed texts. Similarly, Gupta et al. (2024a) focuses on multilingual LLMs' performance in code-mixed machine translation tasks. Experimental results suggest that better code-mixed translation quality is obtained from k-shot prompting rather than 0-shot prompting. Unfortunately, while all these existing studies focus on benchmarking LLMs on code-mixed tasks, none of them offer any solutions for improving performance on such tasks.

RLHF in machine translation. RLHF fine-tunes LLMs using human preference data to align outputs with user expectations. Xu et al. (2024) explores modeling translation preferences with RLHF and constructs reward models by contrasting deficiencies in machine translation compared to human translation from published books. He et al. (2024) investigates the possibility of utilizing the quality estimation (QE) model as the reward model to predict human preferences during RLHF. Experiments show that QE-based feedback training is highly data-efficient. Lai et al. (2024) introduces a framework that models hierarchical rewards in RLHF, and tests their approach in long-form question answering and machine translation tasks. They demonstrate how well hierarchical reward modeling works to improve LLM training procedures for greater consistency with human preferences. Unfortunately, prior work in this space focuses solely on monolingual machine translation tasks. In contrast, we focus on code-mixed machine translation.

RLAIF (Reinforcement Learning from AI Feedback). Collecting human preference data at scale for RLHF is expensive and time-consuming. As a workaround, some recent work attempts to replace human feedback with AI (or LLM) feedback, which is then used as preference data to power the conventional RLHF training procedure. Bai et al. (2022) first introduced this RLAIF procedure, where an AI labeler identified harmful or harmless outputs to construct a reward model for policy optimization and model alignment. Lee et al. (2024) focus on RLAIF for text summarization and dialogue generation tasks and show that RLAIF achieves human-level performance. Li et al. (2024) propose phased annotations on different prompt categories during the AI preference labeling process, greatly improving the accuracy of AI annotations, resulting in a more robust helpfulness model. To the best of our knowledge, this paper represents the first attempt at adapting RLAIF to improve the ability of LLMs to handle code-mixed language, as no prior work has addressed this task.

3 CHAI: RLAIF for Code-Mixed Translation

Reinforcement Learning from Human Feedback (RLHF) is a highly popular and effective technique for aligning the output of LLMs with human-specified preferences (Ouyang et al., 2022). Unfortunately, a key obstacle prohibiting the large-scale use of RLHF is that the quality of the reward model (a key component of RLHF used to fine-tune the final policy model) highly depends on access to high-quality human preference labels. Collecting these preference labels at scale from human annotators is expensive and time-consuming.

To address this issue, recent work (Bai et al., 2022) has proposed replacing human annotators with AI (more specifically, LLM) annotators to efficiently generate preference label data at scale, which can then be used to train the reward model (inside a conventional RLHF pipeline). This novel paradigm of aligning LLMs with (desirable) preferences is called Reinforcement Learning from AI Feedback (RLAIF) (Lee et al., 2024), and it has been successfully adopted to achieve model alignment across various use cases, such as reducing harmful outputs (Li et al., 2024), etc.

In this section, we propose CHAI, a novel general-purpose RLAIF framework to improve the ability of multilingual LLMs to handle code-mixed language. To the best of our knowledge, CHAI is the first to apply RLAIF (or RLHF) to improve model alignment for code-mixed use cases. Specifically, CHAI focuses on using RLAIF to improve LLMs' alignment on the task of code-mixed translation (i.e. translating monolingual text to code-mixed text) using AI-annotated preference labels. Next, we describe CHAI's overall architecture (see Figure 1).

Base LLM Model. The RLAIF procedure starts by using an existing off-the-shelf LLM as a base model (referred to as Base-LLM or π^{base} in Figure 1), which is then further optimized (or aligned) using the RLAIF procedure. In CHAI, we use Llama-3.1-8B-Instruct (Grattafiori and et. al., 2024) as our base model, as (i) it is a robust multilingual LLM (with support for English, Hindi, German, French, and Italian, among others); and (ii) it has demonstrated strong performance in machine translation tasks (Xu et al., 2023), the primary task of interest in this paper, making it an ideal choice for an RLAIF-driven code-mixed translation pipeline.

Stage 1: Supervised Fine Tuning of Base Model Next, we use the base model and conduct supervised fine-tuning on it using domain-specific data (for code-mixed translation) to adapt the base LLM to the target task (of translating monolingual text into code-mixed text). More formally, given a parallel corpus $\mathcal{D}_{\text{parallel}} = \{(x^{(i)}, y^{(i)})\}_{i=1,\ldots,n}$ where x_i represents the source (English) sentences, and y_i represents the corresponding (code-mixed) translation, we apply a fixed prompt template $\mathcal{I}$ (see Appendix A1) on a portion of this parallel corpus and convert it into a training set $\mathcal{D}_{\text{sft}} = \{(\mathcal{I}(x^{(i)}), y^{(i)})\}_{i=1,\ldots,n}$ that can be used to fine-tune our Llama-3.1-8B-Instruct base model. In particular, π^{base} is supervised fine-tuned (SFT) using a next-token prediction objective on this training set $\mathcal{D}_{\text{sft}}$ (Radford et al., 2019). This SFT version of the base model is referred to as SFT-LLM or π^{sft} in Figure 1 (and in the rest of the paper).

Given the widespread prevalence of code-mixed language usage in India (in the form of Hinglish, or Hindi+English) (Thara and Poornachandran, 2018), we focus on using datasets for English $\rightarrow$ Hinglish translation in CHAI to power this SFT stage. In particular, we utilize the following two datasets and use it as our parallel corpus $\mathcal{D}_{\text{parallel}}$:

- **MixMT 2022 shared task** (Srivastava and Singh, 2022), which contains $\sim$1800 parallel English sentences along with multiple human-generated Hinglish translations.
- **ALL-CS dataset** (Tarunesh et al., 2021), which contains 9290 English sentences and multiple Hinglish translations for each sentence (only movie subset is included).

For each of these datasets, we first pair each English sentence with each of the available Hinglish translations, and this results in a total of 3873 data points (from the MixMT dataset) + 11317 data points (from the All-CS dataset) = 15190 datapoints inside our parallel corpus $\mathcal{D}_{\text{parallel}}$, a portion of which is then converted into the $\mathcal{D}_{\text{sft}}$ dataset (as explained above).

Stage 2: Reward Model Training using AI Feedback. The key distinguishing characteristic of an RLAIF framework is that we use an AI or LLM model (instead of a human annotator) to annotate preference data. Once generated, this preference

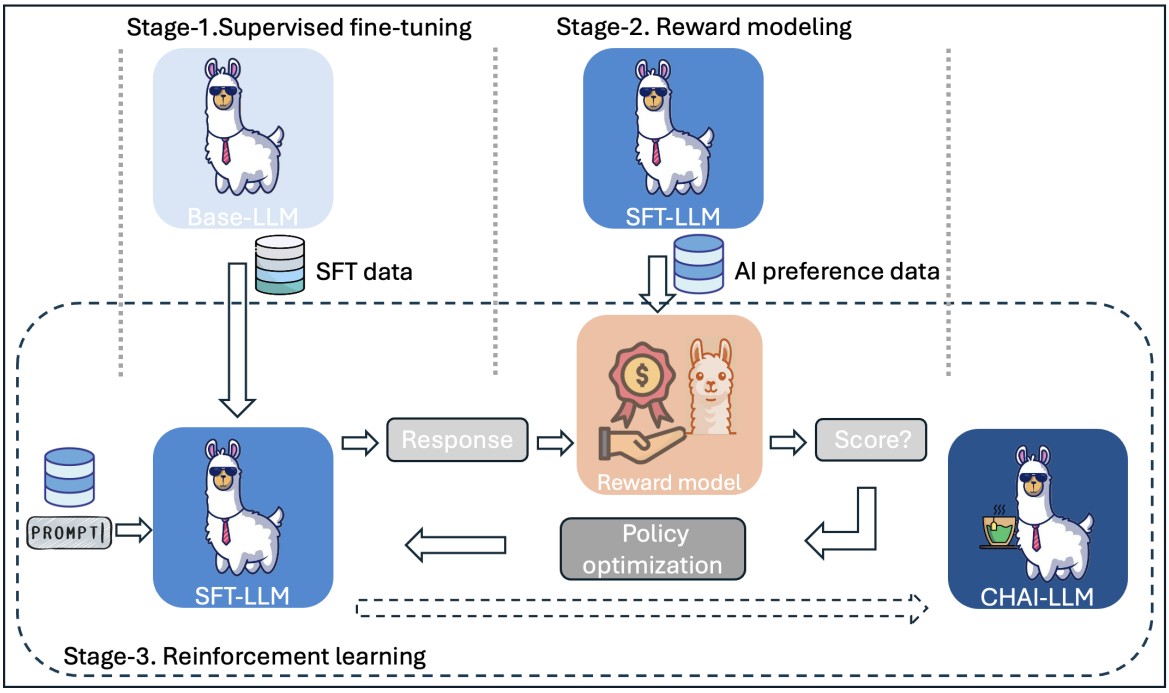

Figure 1: Overall architecture of the RLAIF Procedure used in CHAI.

data is used to train a reward model, and the rest of the RLAIF pipeline mimics the steps in RLHF. We now explain how this is accomplished in CHAI for the task of code-mixed translation.

2.1 Collecting Preference Data Using LLMs We use a portion of the $\mathcal{D}_{\text{parallel}}$ corpus (from Stage 1) and convert it into a preference dataset as follows: (i) each source (English) sentence is paired with two alternative Hinglish translations; (ii) these three sentences are fed into a prompt template $\mathcal{I}_{pref}$ (see Appendix A.5) that generates a custom prompt for an LLM annotator asking it to select which of the two provided Hinglish sentences is a better code-mixed translation for the source English sentence. To mitigate positional bias (Pezeshkpour and Hruschka, 2023; Li et al., 2024) in preference labeling of code-mixed text, we randomly switch the position of the two candidate Hinglish translations before presenting them to the LLM annotator (see Appendix A.2 for more details on positional bias).

Our final preference dataset contains 15190 distinct prompts (of type $\mathcal{I}_{pref}$) that can be passed to an LLM annotator to get a preference label. CHAI uses GPT-4o (OpenAI et al., 2024) [2] as an LLM annotator, each prompt is passed to GPT-4o at three different temperature settings (T=0.1, 0.3, 0.5) to get three preference labels, and the final binary preference label (Y=0 or 1 means that the LLM annotator prefers the first or second code-mixed translation, respectively) is obtained through a majority vote on these three labels. To the best of our knowledge, this represents the first-ever attempt at utilizing LLM annotation abilities for annotating tasks related to code-mixing.

2.2 Reward Model Training This LLM-annotated preference label dataset is used to train a reward model (a key component in the RLAIF framework), which outputs numerical scores in response to LLM generated responses provided as input. Intuitively, the trained reward model should be such that LLM responses that are closely (or weakly) aligned with AI preferences (expressed in our preference dataset) should receive high (or low) scores from the reward model.

In CHAI, we train our reward model as follows: (i) we take π^{sft} (our SFT model from Stage 1) and change its last neuronal layer from a language modeling head (i.e., output logit of each token in vocabulary) into a linear layer which generates a singular scalar prediction representing the output reward score. (ii) To get the final reward model, this modified version of π^{sft} is trained on the LLM-annotated preference dataset using the Bradley-Terry model (Bradley and Terry, 1952), which provides a functional form for the probability that for an English sentence x, the LLM labeler

[2] GPT-4o points to gpt-4o-2024-11-20

prefers its chosen Hinglish translation y_c over the rejected translation y_r:

$$P\{i \succ j\} = \frac{e^{r(x,y_c)}}{e^{r(x,y_c)} + e^{r(x,y_r)}} \quad (1)$$

where $r(x, y_c)$ and $r(x, y_r)$ denote the reward model scores for the chosen and rejected Hinglish translations, respectively. Finally, this probability is incorporated into a negative log-likelihood loss function as follows:

$$\mathcal{L}(r) = -\mathbb{E}_{\mathcal{D}_{\text{rm}}}[\log P\{i \succ j\}] \quad (2)$$

where $\mathcal{D}_{\text{rm}} = \{x^{(i)}, y_{\text{c}}^{(i)}, y_{\text{r}}^{(i)}\}_{i=1}^{N}$ represents the preference labeled dataset for all X data points annotated by the LLM.

Stage 3: Tuning Policy Model with Reinforcement Learning. Finally, we train a policy model π^{rl} (initialized from π^{sft}) to maximize the expected score returned from the reward model using general-purpose reinforcement learning algorithms, such as proximal policy optimization (PPO) (Schulman et al., 2017). More precisely, we optimize the policy model π^{rl} to maximize this objective function:

$$r_{total} = r(x, y) - \eta KL(\pi^{\text{rl}}(y|x)||\pi^{\text{sft}}(y|x)) \quad (3)$$

where r refers to the reward score based on a single sample, and the KL divergence term (i) acts as an entropy bonus, preserving generation diversity and preventing pattern-collapse into singular high-reward responses (Jaques et al., 2019); while (ii) also ensuring that the RL policy's output does not deviate drastically from the distribution where the reward model is accurate (Laidlaw et al., 2024; Wang et al., 2024). Finally, η is a coefficient that trades-off the two terms in this objective function.

4 Experimental Evaluation

We primarily focus our experimental evaluation on analyzing the effectiveness of CHAI in improving the ability of our base Llama-3.1-8B-Instruct model on the task of English → Hinglish translation. Note that while our CHAI framework is general enough to handle code-mixed translation tasks for any language pair, we focus our evaluation to English → Hinglish because there are very few large-scale datasets similar to MixMT 2022 and All-CS available in other language pairs. In particular, MixMT 2022 and All-CS contain multiple target Hinglish translations for every source English sentence, and these multiple target translations are crucial in enabling LLMs to provide preference labels in Stage 2 of the CHAI framework. As such, we leave exploration of other language pairs to future work, especially given the non-trivial effort in collecting such data in other language pairs using human annotators. Nevertheless, we do provide an analysis of the cross-lingual transfer ability of our CHAI-powered LLM (trained specifically for English → Hinglish translation) on additional language pairs (in Table 4).

Evaluation Metrics. To understand the impact of CHAI on the quality of code-mixed translation, we utilize five well-studied metrics: (i) *chrF* (Popović, 2015), which calculates a character n-gram F-score based on the overlap between predicted and reference sentences; (ii) *chrF++* (Popović, 2017), which improves correlations with human assessment by adding word unigrams and bigrams to the standard chrF score; (iii) COMET (Rei et al., 2020), which generates embeddings of the source, hypothesis, and reference sentences with a cross-lingual encoder (Conneau, 2019), and predicts the score of the given translation[3]. To validate the impact of CHAI on classification tasks (especially the sentiment analysis task), we use two classic metrics: (i) classification accuracy; (ii) weighted F1-score.

In addition to these classical evaluation metrics, we also utilize human and LLM evaluators to calculate the win rate (Lee et al., 2024). (iv) To compute win rate with human evaluators, three human evaluators[4] fluent in both English and Hindi were recruited. For each source English sentence in the test set (of MixMT 2022), we generated two Hinglish translations, one using the CHAI-powered LLM and the other using the base LLM (π^{base}). These two Hinglish translations were shown (in random order) to each human evaluator, who were asked to select their preferred translation of the source English sentence. A majority vote was used to determine the evaluators' aggregate preference label. (v) Similarly, to calculate win rate with LLM evaluators, we generated two Hinglish translations for each test data point (as described above) and presented them in random order to a Gemini-1.5-Flash-001 (Team et al., 2024) model across three different temperature settings (T=0.1, 0.3, 0.5), and aggregated results using a majority vote. In both

[3]We use reference-based evaluation model wmt22-comet-da to calculate the COMET score.

[4]All our study protocols were approved by an Institutional Review Board

Prompt	Alignment score
Basic 0-shot	60.30%
Basic + rule 0-shot	**61.8%**
Basic 1-shot	57.70%
Basic 2-shot	54.90%
Basic 3-shot	56.50%
Basic + rule 1-shot	59.70%
Basic + rule 2-shot	55.40%
Basic + rule 3-shot	57.60%
Basic + CoT 0-shot	56.40%
Basic + rule + CoT 0-shot	59.40%
Basic + rule + CoT 1-shot	58.90%
Basic + rule + CoT 2-shot	60.20%

Table 1: Alignment scores between human annotators and LLM annotators utilizing different prompting strategies.

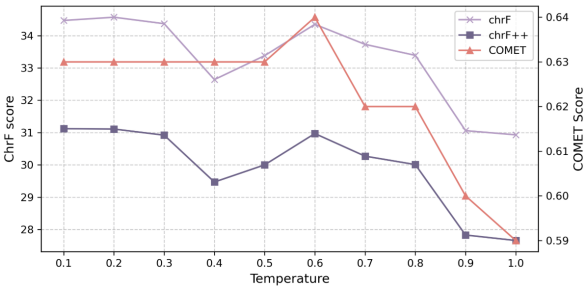

Figure 2: Relationship between the temperature and the quality of code-mixed machine translation.

cases, the *win rate was defined as the proportion of test data points for which the Hinglish translation generated by our CHAI-powered LLM was preferred by the evaluators over the Hinglish translation generated by the base LLM*.

Evaluation Datasets. All machine translation experiments are evaluated on the test set of MixMT 2022 shared task (Srivastava and Singh, 2022). The experiments on cross-lingual transfer ability rely on English → Bengali+English, English → French+English, and English → Spanish+English corpora contained in (Gupta et al., 2024c). The sentiment analysis experiments are evaluated based on the whole dataset of SentMix-3L (Raihan et al., 2023) and the test set of SemEval-2020 Task 9 (Patwa et al., 2020).

We now present results in three stages. First, we present results analyzing the ability of LLM annotators to mimic human preferences in code-mixed translation tasks. We also present results of fine-tuning several hyperparameters in the CHAI framework. Second, we present our main evaluation result of comparing code-mixed translation quality of CHAI-powered LLMs against state-of-the-art open-source LLMs to understand its effectiveness. Finally, we present results analyzing transfer learning abilities of CHAI powered LLMs by evaluating its performance on Hinglish sentiment analysis & cross-lingual machine translation tasks.

LLM Annotator Alignment. To generate preference labels via LLM annotators in Stage 2 of the CHAI framework, we compared the preference labels generated via several permutations and combinations of three different types of prompting strategies (basic prompting A3, rule-augmented prompting A4, and chain-of-thought prompting A5) against human-annotated preferences (three independent human-annotators were also used to provide preference labels on training data points). Table 1 lists the alignment scores (defined as the fraction of training data points on which the LLM annotation matched the human-generated annotation) achieved by LLM annotators powered by different prompting strategies. This figure shows that basic prompting with specified preference annotation rules for code-mixed texts outperforms all other strategies by 1.5% (on average) and achieves the highest alignment score of 61.8%. In particular, this table shows that having additional rules in the prompt helps improve the alignment of LLM annotators (1.28% increase in alignment score on average) on code-mixed translation tasks. Surprisingly, Table 1 shows that chain-of-thought (CoT) prompting and k-shot prompting fails to improve alignment in code-mixed scenarios, possibly because of inconsistencies in grammatical structure of code-mixed texts leads CoT and k-shot prompting astray. In the rest of the experiments, we fix our prompting strategy to the best-performing strategy in Table 1.

Impact of Supervised Fine Tuning. We conduct an ablation study to evaluate the impact of supervised fine-tuning (SFT) in Stage 1 of the RLAIF framework on code-mixed translation. Table 2 compares the quality of code-mixed translation generated with the standard RLAIF framework (which includes the SFT step) and the translation generated with a version of RLAIF in which no SFT training is done in Stage 1. Both human and Gemini evaluators prefer RLAIF (no SFT) over standard RLAIF, with win rates of 55.47% and 63.30%, respectively (Table 2). Results with conventional metrics show similar trends. These results show

Evaluator	Results	En-> Hinglish
Gemini	RLAIF	36.70%
	RLAIF(no SFT)	63.30%
Human	RLAIF	44.53%
	RLAIF(no SFT)	55.47%
chrF	RLAIF	42.09
	RLAIF(no SFT)	42.43
chrF++	RLAIF	38.01
	RLAIF(no SFT)	38.04
COMET	RLAIF	0.67
	RLAIF(no SFT)	0.70

Table 2: Performance of RLAIF with (without) SFT.

that using SFT is counterproductive in our context, lowering the code-mixed translation quality. In part, these results could also be explained by our choice of an instruction-tuned model (Llama-3.1-8b-Instruct) as our base model. As instruction-tuned models have undergone one round of SFT during their training phase, the additional SFT step in the standard RLAIF framework may have led to overfitting, reducing the model's generalizability. Future research should investigate alternative fine-tuning strategies or data augmentation techniques to enhance generalization without compromising translation quality. Thus, all future CHAI experiments exclude SFT.

Tuning LLM Temperature. In Figure 2, we compare the variation in code-mixed translation quality (as measured by chrF, chrF++, and COMET on Y-axes) with increasing values of temperature for the CHAI-powered LLM (X-axis). This figure shows that all three metrics are optimized at T=0.6. Thus, we fix the temperature of the CHAI-powered LLM to T=0.6 in all future experiments.

Impact of CHAI on Translation Quality. Having identified the best prompting strategy, temperature, etc., we now train a CHAI-powered LLM with these optimal hyperparameters to evaluate its effectiveness in improving the quality of code-mixed translation. Table 3 compares the quality of code-mixed translation generated by the CHAI-powered LLM against the translations generated by the base model (π^{base}). This table shows that the win rate achieved by translations generated by the CHAI-powered LLM outperforms the win rate (of π^{base}) by 13.42% (for LLM evaluators) and 25.66% (for human evaluators). Similarly, CHAI-powered LLM outperforms π^{base} by achieving 27.57% higher chrF, 27.16% higher chrF++, and 10.93% higher COMET scores. In a nutshell, these results establish that the CHAI framework is highly successful at improving the ability of LLMs to handle code-mixed translation tasks.

Moreover, examples shown in Table 5 compare the translations generated from π^{base} and CHAI-powered LLM on two sampled data points, which are representative of the general trend. From both samples, it is evident that code-mixed translations generated by the CHAI-powered LLM express more accurate and natural-sounding language, which aligns well with human preferences.

Evaluator	Results	En -> Hinglish
Gemini	π^{base}	43.29%
	CHAI-LLM	56.71%
Human	π^{base}	37.17%
	CHAI-LLM	62.83%
chrF	π^{base}	33.77
	CHAI-LLM	43.08
chrF++	π^{base}	30.49
	CHAI-LLM	38.77
COMET	π^{base}	0.64
	CHAI-LLM	0.71

Table 3: Measuring CHAI's ability in improving code mixed translation ability.

Cross-lingual Transferability. Next, we examine if translation preferences learned during post-training (especially, the RLAIF procedure) enhance cross-lingual transfer. Three translation directions: (i) English → English + Bengali;(ii) English → English + French; and (iii) English → English + Spanish are evaluated in Table 4, which shows that our RLAIF procedure has indeed improved the cross-lingual transfer ability of our CHAI-powered LLM (as compared to SFT-LLM) on at least two out of three language pairs (English → English + Bengali & English → English + Spanish). This results mirrors existing findings showing cross-lingual transfer ability of LLMs achieved via machine translation tasks (Lample and Conneau, 2019).

Ability to Understand Code-Mixing. Finally, we explore if using RLAIF for code-mixed translation improves an LLM's general ability to handle additional code-mixed tasks. Table 6 compares the accuracy and F1 achieved by our CHAI-powered LLM and the base LLM (π^{base}) on two code-mixed sentiment analysis datasets containing Hinglish sentences as input, and a ternary sentiment (positive,

Original Translation Direction	Evaluator	Results	Translation Direction		
			En→CM of Be and En	En→CM of Fr and En	En→CM of Es and En
En→Hinglish	Gemini	π^{base} Win	49.89%	54.44%	42.86%
		CHAI-LLM Win	50.11%	45.56%	57.14%
	chrF	π^{base}	12.75	34.88	32.72
		CHAI-LLM	19.94	22.07	35.56
	chrF++	π^{base}	11.42	31.52	30.56
		CHAI-LLM	17.48	19.85	33.09
	COMET	π^{base}	0.59	0.67	0.65
		CHAI-LLM	0.66	0.71	0.79

Table 4: Cross-lingual transfer result based on different code-mixed language pairs.

	Results	English -> Hinglish
Sample-1	Input (English)	You can see a gleam in their eye.
	π^{base} output (Hinglish)	Aapko unke aankhon mein ek chhupi hui chot dikh rahi hai.
	CHAI output (Hinglish)	Arre, aapko unke aankhon mein ek gleam dikh raha hai.
	Comments	In the CHAI output, "gleam" remains unchanged, while the rest of the sentence is translated into Hindi. However, in the π^{base} output, "gleam" is mistranslated as "chhupi hui chot" ('hidden injury' in Hinglish), incorrectly translating "gleam" into 'injury', and also adding an unintended descriptor 'hidden'.
Sample-2	Input (English)	Get our egotism out of the way.
	π^{base} output (Hinglish)	Aapke aap mein khelna band kar dena hai.
	CHAI output (Hinglish)	Arre, humari egotism ko aside kar do.
	Comments	π^{base} output misinterprets 'egotism' literally (psychologically) where the translation means "we have to stop playing amongst ourselves", which is unrelated to the given sentence. Instead, CHAI preserves the original meaning.

Table 5: Comparing the translations generated from π^{base} and the CHAI-powered LLM.

neutral, negative) label. This table shows that our CHAI-powered LLM outperforms π^{base} by 14.12% (and 25.64%) on average in terms of accuracy (and F1), which indicates that using RLAIF improves an LLM's ability to handle other code-mixed tasks.

Dataset	LLM	Accuracy	F1_score
SemEval-2020	π^{base}	35.40%	22.65%
	CHAI	36.77%	25.04%
SentMix-3L	π^{base}	44.39%	32.96%
	CHAI	55.21%	46.38%

Table 6: Performance of CHAI on sentiment analysis.

5 Conclusion

This paper introduces CHAI (**C**ode Mixed Understanding via **H**ybrid **AI** **I**nstruction), a novel framework utilizing RLAIF to handle code-mixed language, specifically for machine translation. CHAI provides a cost-effective preference labeling strategy using high-quality open-source datasets and AI labeling. We demonstrate that LLM-as-annotators can effectively annotate code-mixed texts, reducing human annotation costs. Experimental results show CHAI-powered models outperform state-of-the-art open-source LLMs by 25.66% and exhibit cross-lingual transfer in other code-mixed languages.

6 Limitations

Due to the non-trivial effort involved in gathering annotations from professional crowd (human) annotators across different language pairs, this study focuses on a single language pair (Hindi and English) and leave the exploration of other language pairs for future work. This naturally limits our evaluation somewhat. Additionally, the study focuses on implementing CHAI on only one 8-billion parameter version of an open-source LLM (Llama-3.1-8B-Instruct). Extending this study to multiple LLMs is an interesting direction for future work (albeit an expensive one). Next, the study mainly focuses on a single NLP task: machine translation (except for experiments in Table 6). In future work, we aim to experiment with other directionalities of translation and more general NLP tasks such as code-mixed summarization, word-level language identification, etc. Finally, while we recognize that there are other important dimensions for evaluating translation quality such as the presence/absence of bias, helpfulness/harmfulness of translations, etc., this study evaluates performance solely based on translation accuracy. We leave the exploration of these other evaluation dimensions for future work.

7 Ethical Considerations

The problem studied in this paper - development of LLMs for code mixed translation - presents several ethical challenges that need to be discussed and contemplated. First, it is important that such code-mixed LLMs output fair and unbiased translation outputs. In particular, is is necessary to be vigilant about situations in which biases in code-mixed training data lead to biased or skewed translations that may end up reinforcing problematic social norms, or misrepresenting cultural nuances. Additionally, preserving the intent and sentiment of speakers is essential, particularly in settings where such code-mixed translations are used to interact with code-mixed speakers.

Perhaps most importantly, the ethics of circumventing human feedback with AI feedback (as is the norm in RLAIF procedures) needs to be discussed carefully. On the one hand, as the results of this paper show, leveraging AI feedback in RLAIF procedures will speed up the developmennt of inclusive code-mixed LLMs which will help bridge the digital divide, by making the benefits of LLMs available to lots of code-mixed speakers from places like South Asia. On the other hand, utilizing AI feedback (in RLAIF) might mean fewer opportunities for human crowd workers (a majority of whom live in South Asia) to provide annotations and receive renumeration in return. Thus, the ethics of leveraging LLMs as annotators deserves serious discussion (especially with regards to the associated negative impacts on the livelihoods of human crowd annotators).

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Szu-Wei Cheng, Chung-Wen Chang, Wan-Jung Chang, Hao-Wei Wang, Chih-Sung Liang, Taishiro Kishimoto, Jane Pei-Chen Chang, John S Kuo, and Kuan-Pin Su. 2023. The now and future of chatgpt and gpt in psychiatry. *Psychiatry and clinical neurosciences*, 77(11):592–596.

A Conneau. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Aaron Grattafiori and et. al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ayushman Gupta, Akhil Bhogal, and Kripabandhu Ghosh. 2024a. Code-mixer ya nahi: Novel approaches to measuring multilingual llms' code-mixing capabilities. *Preprint*, arXiv:2410.11079.

Ayushman Gupta, Akhil Bhogal, and Kripabandhu Ghosh. 2024b. Code-mixer ya nahi: Novel approaches to measuring multilingual llms' code-mixing capabilities. *arXiv preprint arXiv:2410.11079*.

Ayushman Gupta, Akhil Bhogal, and Kripabandhu Ghosh. 2024c. Multilingual controlled generation and gold-standard-agnostic evaluation of code-mixed sentences. *arXiv preprint arXiv:2410.10580*.

Zhiwei He, Xing Wang, Wenxiang Jiao, Zhuosheng Zhang, Rui Wang, Shuming Shi, and Zhaopeng Tu. 2024. Improving machine translation with human feedback: An exploration of quality estimation as a reward model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8164–8180, Mexico City, Mexico. Association for Computational Linguistics.

Krystal Hu. 2023. Chatgpt sets record for fastest-growing user base - analyst note.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*.

Yuhang Lai, Siyuan Wang, Shujun Liu, Xuanjing Huang, and Zhongyu Wei. 2024. ALaRM: Align language models via hierarchical rewards modeling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7817–7831, Bangkok, Thailand. Association for Computational Linguistics.

Cassidy Laidlaw, Shivam Singhal, and Anca Dragan. 2024. Preventing reward hacking with occupancy measure regularization. *arXiv preprint arXiv:2403.03185*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *Preprint*, arXiv:2309.00267.

Ang Li, Qiugen Xiao, Peng Cao, Jian Tang, Yi Yuan, Zijie Zhao, Xiaoyuan Chen, Liang Zhang, Xiangyang Li, Kaitong Yang, Weidong Guo, Yukang Gan, Xu Yu, Daniell Wang, and Ying Shan. 2024. Hrlaif: Improvements in helpfulness and harmlessness in open-domain reinforcement learning from ai feedback. *Preprint*, arXiv:2403.08309.

Alexandre Magueresse, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges. *arXiv preprint arXiv:2006.07264*.

OpenAI, Josh Achiam, and et. al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Yang Jeong Park, Daniel Kaplan, Zhichu Ren, Chia-Wei Hsu, Changhao Li, Haowei Xu, Sipei Li, and Ju Li. 2024. Can chatgpt be used to generate scientific hypotheses? *Journal of Materiomics*, 10(3):578–584.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. *Preprint*, arXiv:2008.04277.

Pouya Pezeshkpour and Estevam Hruschka. 2023. Large language models sensitivity to the order of options in multiple-choice questions. *arXiv preprint arXiv:2308.11483*.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395.

Maja Popović. 2017. chrf++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, pages 612–618.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Md Nishat Raihan, Dhiman Goswami, Antara Mahmud, Antonios Anstasopoulos, and Marcos Zampieri. 2023. Sentmix-3l: A bangla-english-hindi code-mixed dataset for sentiment analysis. *arXiv preprint arXiv:2310.18023*.

Muhammad Ramzan, Aamir Aziz, and Maimoona Ghaffar. 2021. A study of code-mixing and code-switching (urdu and punjabi) in children's early speech. *Journal of Language and Linguistic Studies*, 17(2):869–881.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation. *arXiv preprint arXiv:2009.09025*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Vivek Srivastava and Mayank Singh. 2022. Overview and results of MixMT shared-task at WMT 2022. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 806–811, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. 2021. From machine translation to code-switching: Generating high-quality code-switched text. *arXiv preprint arXiv:2107.06483*.

Gemini Team, Petko Georgiev, and et. al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

S Thara and Prabaharan Poornachandran. 2018. Code-mixing: A brief survey. In *2018 International conference on advances in computing, communications and informatics (ICACCI)*, pages 2382–2388. IEEE.

Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawendé F Bissyandé. 2023. Is chatgpt the ultimate programming assistant–how far is it? *arXiv preprint arXiv:2304.11938*.

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, et al. 2024. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2023. A paradigm shift in machine translation: Boosting translation performance of large language models. *arXiv preprint arXiv:2309.11674*.

Nuo Xu, Jun Zhao, Can Zu, Sixian Li, Lu Chen, Zhihao Zhang, Rui Zheng, Shihan Dou, Wenjuan Qin, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Advancing translation preference modeling with rlhf: A step towards cost-effective solution. *Preprint*, arXiv:2402.11525.

Ruochen Zhang, Samuel Cahyawijaya, Jan Christian Blaise Cruz, and Alham Fikri Aji. 2023a. Multilingual large language models are not (yet) code-switchers. *arXiv preprint arXiv:2305.14235*.

Ruochen Zhang, Samuel Cahyawijaya, Jan Christian Blaise Cruz, Genta Winata, and Alham Fikri Aji. 2023b. Multilingual large language models are not (yet) code-switchers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12567–12582, Singapore. Association for Computational Linguistics.

Wenbo Zhang, Hangzhi Guo, Prerna Ranganathan, Jay Patel, Sathyanath Rajasekharan, Nidhi Danayak, Manan Gupta, and Amulya Yadav. 2023c. A continual pre-training approach to tele-triaging pregnant women in kenya. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14620–14627.

A Appendix

A.1 Prompt Template to Create Parallel Corpus

See template at Table A1.

Prompt template $\mathcal{I}$:
Translate this from {Source} to {Target}:
[Source]: {x}
[Target]: {y}

Table A1: Prompt template to create parallel corpus, where 'Source' and 'Target' represent the names of the source language and the target language,respectively.

A.2 Positional Bias in Code-mixed Texts

We use the same test set (previously used for section A.3) to evaluate the positional bias problem in annotating code-mixed texts. For each example in the test set, we ask different LLM labelers to generate preference labels for a pair of candidates through the basic prompt in A3. Then the candidate order presented in the prompt is swapped, and the same LLMs are requested to generate preference labels again. If an LLM favors the same opinion on both the original and reversed order of candidates in the prompt, we consider it to be biased.

In this section, we measure position bias by computing the alignment score between the LLM annotated results and human preference labels. From Table A2, we see that both LLM labelers(GPT-4o and Gemini) shows different alignment score on same preference labeling task. This observation indicates the positional bias of LLM labelers also exists through the preference annotation task on code-mixed texts.

LLM labeler	Alignment score
GPT-4o (default order)	59.7%
GPT-4o (switched order)	54.3%
Gemini (default order)	59.0%
Gemini (switched order)	55.2%

Table A2: Performance of LLM labelers with different positional orders.

A.3 Details of Evaluation Set for Alignment Score Calculation

We downsampled from the training set $\mathcal{D}_{\text{rm}}$ and create a evaluation set containing 1000 data points. Each data point contains one English sentence and two corresponding code-mixed Hinglish translations. Each sample is assessed by three independent human annotators. The human preference labels are obtained through the majority voting of three human annotators' results.

A.4 Training Details of RLAIF Procedure

SFT stage. From the ablation study called Impact of Supervised Fine Tuning, we see that SFT step cannot boost LLM's final performance. Therefore, we do not include the SFT stage in training.

Reward model training stage. The reward model is initialized from LlaMA-3.1-8b-Instruct. The

whole training data are used to form the chosen-rejected pairs with translated results collected from the open-source dataset of code-mixed machine translation tasks. We train 3 epochs with the learning rate of 1.0e-4, warmup ratio of 0.1, and maximum input length of 1024.

RL fine-tuning stage. We use the LlaMA-3.1-8b-Instruct as the initial policy. We reuse the input from the training data during the reward model training phase as queries. During RL fine-tuning, we sample from LLM with a temperature T=0.6 and nucleus sampling top_p=0.9 and limit the maximum of generated length to 512. We train the model with a batch size of 16 and the learning rate of 1.0e-5 for 5 epochs.

A.5 Prompts for Preference Labeling

See different prompt strategies at Table A3, Table A4, Table A5, and Table A6.

A.6 Prompt for LLM-based Evaluation

See LLM evaluation prompt at Table A7.

A.7 Recruitment Details

All three human annotators are recruited from the university using convenience sampling. Each person was given 25 U.S. dollars per hour.

Prompt_text: You are a fluent Hinglish speaker. Fluent Hinglish speakers are able to switch between Hindi and English in the same sentence effortlessly while having a conversation.
You have an English sentence for which you'd like to choose the best Hinglish translation.
The English sentence is: {original_sent};
Translated-sentence-0 is: {first_translation};
Translated-sentence-1 is: {second_translation};

Choose a translated statement that best aligns with how a fluent Hinglish speaker talks. The format of the output should be as follows: "My preference is:",followed by the number 0 or 1 (which signifies the corresponding translated sentence) based on your preference.

Table A3: Basic zero-shot prompt for preference labeling on code-mixed texts.

Prompt_text: A good code-mixed translation seamlessly blends elements of two or more languages while maintaining the original meaning and context. It ensures clarity and fluency in both languages, allowing the message to be easily understood by speakers of all involved languages.
Below we define four evaluation axes for code-mixed translation quality: accuracy, naturalness, syntactic correctness, and Code-switching Correctness.
1.Accuracy: It evaluates how effectively the translated sentence retains the meaning and information of the original sentence, while ensuring the correct usage of code-switched terms. For example, does the translation faithfully reflect the content of the original meaning? Is the key information missing, alternated or repeated in translated sentences? Does the translation introduce the new information which are not covered in original sentences?
2.Naturalness: It assesses how natural and easy to understand the translated sentence is. For example, is the new translation elegant? Does the translated sentence seem difficult to understand, awkward, or contain unnatural phrasing?
3.Syntactic correctness: It considers grammar, syntax, and the seamless integration of code-switching in translated sentences. Are there any grammar or syntax issues in translation? Does code-mixing disrupt the flow of the sentence? Is it somewhat smooth but not perfectly integrated? Or is it smooth and seamless?
4.Code-switching Correctness: It evaluates whether the given sentence is a correct instance of code-switching (CS). Specifically, we define a sentence as a correct CS sentence if it meets the following constraints: (a) it is not entirely in Hindi or English, and (b) no language other than Hindi or English is used.

You are a fluent Hinglish speaker. Fluent Hinglish speakers are able to switch between Hindi and English in the same sentence effortlessly while having a conversation.
You have an English sentence for which you'd like to choose the best Hinglish translation.
The English sentence is: {original_sent};
Translated-sentence-0 is: {first_translation};
Translated-sentence-1 is: {second_translation};

Choose a translated statement that best aligns with how a fluent Hinglish speaker talks. The format of the output should be as follows: "My preference is:",followed by the number 0 or 1 (which signifies the corresponding translated sentence) based on your preference.

Table A4: rule-augmented zero-shot prompt for preference labeling on code-mixed texts.

Prompt-1 (output_rationale): You are a fluent Hinglish speaker. Fluent Hinglish speakers are able to switch between Hindi and English in the same sentence effortlessly while having a conversation.
You have an English sentence and two of its possible Hinglish translation.
Explain the reason that which translation is better.
The format of the output should be as follows: "Rationale:",followed by the reasons in one paragraph.

The English sentence is: {original_sent};
Translated-sentence-0 is: {first_translation};
Translated-sentence-1 is: {second_translation};

Prompt-2 (output_preference): You are a fluent Hinglish speaker. Fluent Hinglish speakers are able to switch between Hindi and English in the same sentence effortlessly while having a conversation.
You have an English sentence, two of its possible Hinglish translation, and corresponding rationale.
Choose a translated statement that best aligns with how a fluent Hinglish speaker talks.
The format of the output should be as follows: "My preference is:",followed by the number 0 or 1 (which signifies the corresponding translated sentence) based on your preference.

The English sentence is: {original_sent};
Translated-sentence-0 is: {first_translation};
Translated-sentence-1 is: {second_translation};
Rationale: {rationale}

Table A5: Basic zero-shot chain-of-thought prompt for preference labeling on code-mixed texts, where we first generate the rational based on prompt-1 and then concatenate it with prompt-2 to generate the final preference label.

Prompt: You are a fluent Hinglish speaker. Fluent Hinglish speakers are able to switch between Hindi and English in the same sentence effortlessly while having a conversation.
You have an English sentence for which you'd like to choose the best Hinglish translation.
Choose a translated statement that best aligns with how a fluent Hinglish speaker talks.
You could only output 0 (if you prefers Translated-sentence-0) or output 1 (if you prefers Translated-sentence-1)

»»»» Example »»»»
The English sentence is: <original_sent for example-1>;
Translated-sentence-0 is: <first_translationfor example-1>;
Translated-sentence-1 is: <second_translation for example-1>;
My preference is: <label for example-1>

»»»» Follow the instructions and the example(s) above »»»»
The English sentence is: {original_sent};
Translated-sentence-0 is: {first_translation};
Translated-sentence-1 is: {second_translation};
My preference is:

Table A6: Basic 1-shot prompt for preference labeling on code-mixed texts.

System_role: You are a translation expert in {source_language}, {target_language}, code-mixing of {source_language} and {target_language}. I need your help in impartially judging the quality of two translations.
Prompt_text: Below we define four evaluation axes for code-mixed translation quality: accuracy, naturalness, syntactic correctness, and Code-switching Correctness.
1.Accuracy: It evaluates how effectively the translated sentence retains the meaning and information of the original sentence, while ensuring the correct usage of code-switched terms. For example, does the translation faithfully reflect the content of the original meaning? Is the key information missing, alternated or repeated in translated sentences? Does the translation introduce the new information which are not covered in original sentences?
2.Naturalness: It assesses how natural and easy to understand the translated sentence is. For example, is the new translation elegant? Does the translated sentence seem difficult to understand, awkward, or contain unnatural phrasing?
3.Syntactic correctness: It considers grammar, syntax, and the seamless integration of code-switching in translated sentences. Are there any grammar or syntax issues in translation? Does code-mixing disrupt the flow of the sentence? Is it somewhat smooth but not perfectly integrated? Or is it smooth and seamless?
4.Code-switching Correctness: It evaluates whether the given sentence is a correct instance of code-switching (CS). Specifically, we define a sentence as a correct CS sentence if it meets the following constraints: (a) it is not entirely in Hindi or English, and (b) no language other than Hindi or English is used.

Next, I will provide you with the original text under the <Original> tag, first translation under the <Translation_1>, and second translation under the <Translation_2>.
Please let me know which one is better according to these criteria. Please give your judgment directly (output "Translation_1" or "Translation_2" only) and do not output additional explanations.
<Original>
{original_sent}
</Original>

<Translation_1>
{first_translation}
</Translation_1>

<Translation_2>
{second_translation}
</Translation_2>

Table A7: Prompt for LLM-based evaluation.