

ARJUN: A Parameter-Efficient Multi-Objective Pre-Training Framework for Language Models

Subhadip Maji Arnab Bhattacharya

Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur
{subhadip,arnabb}@cse.iitk.ac.in

Abstract

The proliferation of large-scale language models has significantly advanced natural language processing capabilities; yet their substantial computational requirements create barriers for deployment in resource-constrained environments and limit accessibility for under-represented language communities. To address this fundamental issue, we present a *novel multi-objective pre-training approach* for developing *parameter-efficient language models*. Our method, implemented in the ARJUN model family, strategically combines four complementary pre-training objectives within a unified transformer encoder-decoder framework: (1) Masked Token Prediction (MT) for bidirectional understanding, (2) Next Token Prediction (NT) for autoregressive generation, (3) Masked Span Prediction (MS) for multi-token dependencies, and (4) Complete Sentence Prediction (CS) to bridge discriminative and generative tasks. Using this approach, our Arjun-Base model (69M parameters) consistently outperforms standard baselines 2.5–4× larger. Extending this architecture, our scaled-up Arjun-Large variant (up to 256M parameters) demonstrates superior architectural efficiency, establishing strong performance baselines within matched parameter budgets. Evaluations across 20 discriminative and generative tasks in Bangla, Tamil, and Hindi show prominent gains, including up to +6.5 percentage points (pp) over comparable baselines on macro-F1 scores and +17.5 pp sacreBLEU on generation. Finally, ablation studies confirm that joint training on these four objectives yields synergistic representations that consistently outperform single-task configurations across all scales.

1 Introduction

The rapid advancement of large pre-trained language models has revolutionized natural language processing, with models such as BERT, GPT, and

T5 achieving remarkable performance across diverse tasks (Devlin et al., 2018; Raffel et al., 2019). However, these models require billions of parameters, thereby creating substantial barriers for deployment in resource-constrained environments. Recent research has explored parameter-efficient approaches, including adapter-based methods, prefix tuning, and low-rank adaptation (Hu et al., 2021), to address these limitations. While these techniques reduce the computational overhead of fine-tuning large models, they still rely on massive pre-trained backbones and do not fundamentally address the efficiency challenges at the *pre-training stage*. Simultaneously, multi-objective pre-training has emerged as a promising paradigm for developing more versatile language representations (Ye et al., 2024; Doddapaneni et al., 2022). Studies (Tay et al., 2022; Ye et al., 2024) have shown that combining complementary training objectives can lead to models that better capture both local and global linguistic patterns, and result in improved generalization across downstream tasks.

Building on these insights, we propose a novel multi-objective pre-training framework that achieves strong performance with significantly fewer parameters. Our approach centers on the joint optimization of four carefully designed pre-training objectives, each targeting different aspects of language understanding. Figure 1 illustrates our overall architecture, showing how these four objectives: Masked Token Prediction (MT), Next Token Prediction (NT), Masked Span Prediction (MS), and Complete Sentence Prediction (CS), interact within a unified transformer encoder-decoder framework. The MT objective, inspired by BERT’s masked language modeling, enables bidirectional context understanding (Devlin et al., 2018). The NT objective, following GPT’s autoregressive approach (Radford et al., 2018), captures causal dependencies and generation capabilities. The MS and CS objectives, adapted from T5’s (Raffel et al.,

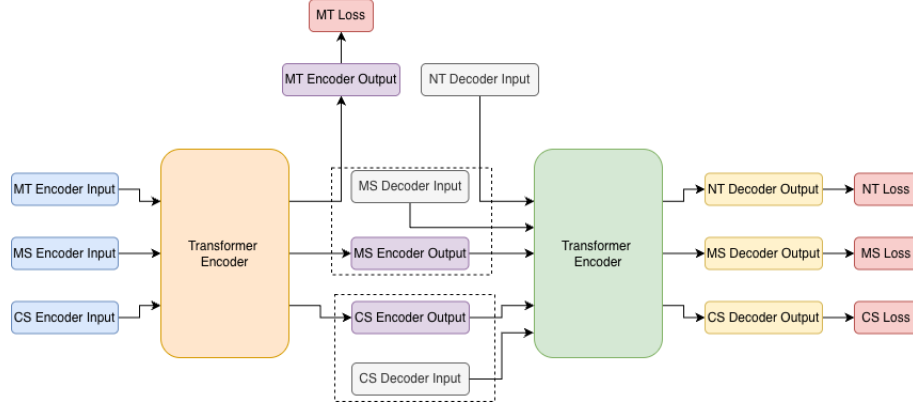


Figure 1: Overall architecture of the Arjun model showing how the four pre-training objectives (MT, NT, MS, and CS) interact with the transformer encoder-decoder components. Arjun is trained with MT, MS, CS and NT losses jointly optimized. (Eq. (5) represents the joint loss. Table 1 shows examples.)

2019) span corruption strategy, teaches the model to handle variable-length semantic units that bridge discriminative and generative tasks. The four objectives are complementary to each other – while MT provides bidirectional contextual understanding, NT teaches sequential generation patterns, MS captures multi-token semantic dependencies, and CS develops sentence-level coherence.

We implement our framework in **Arjun**, a family of transformer encoder-decoder models spanning Small, Base, and Large configurations (up to 256M total parameters). Despite their relatively modest sizes, Arjun models consistently achieve competitive or superior performance compared to much ($2\text{--}4\times$) larger models across 20 diverse discriminative and generative tasks in Bangla, Tamil, and Hindi. Notably, they demonstrate highly competitive accuracy and in several cases outperform strong contemporary baselines like ModernBERT (Warner et al., 2024), IndicBERT-v2 variants (278M) (Doddapaneni et al., 2022), and BanglaByT5 (Bhattacharyya and Bhattacharya, 2025a), while requiring significantly fewer parameters.

The contributions of this work are three-fold:

1. We introduce a novel multi-objective pre-training framework for language models that strategically combines four complementary objectives to create highly effective and parameter-efficient representations.¹
2. We demonstrate through a comprehensive evaluation that joint optimization of these objectives yields superior performance compared to single-objective approaches, with our Base

model (69M) outperforming baselines up to $4\times$ larger, and our Large variant achieving highly competitive performance that validates our architectural efficiency.

3. We provide a thorough analysis through ablation studies showing that each component contributes meaningfully to the overall performance, thus validating the synergistic design choices of our multi-objective approach.

2 Related Work

The landscape of pre-trained language models has been shaped by several foundational approaches and subsequent innovations. BERT introduced masked language modeling as a powerful pre-training objective, achieving bidirectional context understanding through its encoder-only architecture (Devlin et al., 2018). RoBERTa improved upon BERT by removing the next sentence prediction objective, using dynamic masking, and training with larger batch sizes and more data, demonstrating that careful optimization of training procedures can significantly enhance performance (Liu et al., 2019). The T5 framework revolutionized the field by casting all NLP tasks into a unified text-to-text format, employing span corruption as its primary pre-training objective within an encoder-decoder architecture (Raffel et al., 2019). Building on autoregressive foundations, XLNet proposed permutation language modeling to combine the benefits of autoregressive and autoencoding approaches, achieving bidirectional context modeling without relying on masked tokens (Yang et al., 2019). Parameter efficiency became a critical concern with ALBERT, which introduced factorized embedding param-

¹The code is available at <https://github.com/pidahbus/multi-task-learning>.

terization and cross-layer parameter sharing to reduce BERT’s parameter count by up to 89% while maintaining competitive performance (Lan et al., 2019). Complementing these parameter-efficient approaches, the BabyLM challenge (Warstadt et al., 2025) has recently spurred research into data efficiency by benchmarking models trained on strictly constrained corpora, highlighting the viability of smaller-scale pre-training. Several works have explored sophisticated multi-objective frameworks (Ye et al., 2024): ELECTRA introduced replaced token detection as an alternative to masked language modeling, training models to distinguish real tokens from generated replacements (Clark et al., 2020), while UL2 proposed a unified framework combining multiple denoising objectives through mixture-of-denoisers, demonstrating universal effectiveness across diverse NLP tasks and achieving state-of-the-art results on over 50 benchmarks (Tay et al., 2022). GLM presented autoregressive blank infilling with 2D positional encodings, successfully unifying natural language understanding and generation capabilities within a single pre-training framework (Du et al., 2022). For Indian languages specifically, the AI4Bharat (Kakwani et al., 2020) initiative has contributed significantly with IndicBERT models trained on comprehensive Indian language corpora, while MuRIL has targeted multilingual Indian language understanding (Khanuja et al., 2021). Our work builds upon these advances by proposing a novel multi-objective pre-training approach that strategically combines complementary objectives to achieve strong performance with exceptional parameter efficiency. Furthermore, while mT5 (Xue et al., 2021) and XLM-R (Conneau et al., 2020) have established strong paradigms for multilingual representation, they predominantly rely on singular pre-training objectives (span corruption and masked language modeling, respectively). Recent literature also highlights the impact of specialized masking strategies, such as contiguous span masking (Joshi et al., 2019). While our framework currently utilizes standard random masking to ensure a strictly controlled architectural comparison, integrating these targeted masking strategies remains a highly promising avenue for future enhancement.

3 Methodology

Our goal is to pre-train a language model (with parameters θ) to capture rich linguistic and semantic representations by jointly optimizing multiple

training objectives. Figure 1 illustrates the overall architecture of our proposed Arjun model. Let $x = [x_1, x_2, \dots, x_T]$ be an input sequence sampled from the corpus. We define the following 4 pre-training objectives.

3.1 Masked Token Prediction (MT)

Following BERT, we randomly select a subset of token positions $M \subset \{1, 2, \dots, T\}$ for masking, typically 15% of the tokens. We replace the tokens at these positions with a special [MASK] token to create a corrupted sequence $\tilde{x}$.

The objective is to minimize:

$$L_{MT}(\theta, x) = - \sum_{i \in M} \log p_{\theta}(x_i | \tilde{x}) \quad (1)$$

Unlike BERT, we do not implement the Next Sentence Prediction (NSP) objective, as prior experiments (Liu et al., 2019) have demonstrated its insignificant contribution to model performance.

3.2 Next Token Prediction (NT)

Following the GPT approach, this is an autoregressive language modeling objective where the model predicts each token given its preceding context:

$$L_{NT}(\theta, x) = - \sum_{t=1}^T \log p_{\theta}(x_t | x_1, \dots, x_{t-1}) \quad (2)$$

3.3 Masked Span Prediction (MS)

Instead of masking individual tokens, we randomly select spans of consecutive tokens for masking. We mask 15% of the words (not tokens) in the input, with consecutive masks appended to create single spans. Each span is replaced with a unique sentinel token (e.g., [S1], [S2]) to help the model predict spans in the correct order. The objective is:

$$L_{MS}(\theta, x) = - \sum_{s \in S} \log p_{\theta}(x_s | \tilde{x}) \quad (3)$$

where x_s denotes the original sequence of tokens in span s .

3.4 Complete Sentence Prediction (CS)

We pick a token position in a sentence **uniformly at random**. This random sampling strategy forces the model to learn completions from highly variable prefix lengths, rather than memorizing specific syntactic boundaries. On the encoder side, tokens from the beginning of the sentence up to the chosen position are provided as input. On the decoder

side, the model is trained to generate the rest of the sentence:

$$L_{CS}(\theta, x) = - \sum_{t=k+1}^T \log p_{\theta}(x_t | x_1, \dots, x_k) \quad (4)$$

where k is the randomly chosen token position in the sentence.

3.5 Joint Training and Model Architecture

The multi-objective pre-training framework Arjun employs a transformer encoder-decoder architecture that processes different types of inputs depending on the training task, with MT utilizing only the encoder, NT using only the decoder, and MS and CS leveraging both components to develop comprehensive language representations. All the four tasks are jointly trained within each mini-batch of our training process. We randomly split every batch into four equal parts for four different tasks as shown in Table 1.²

Eq. (5) represents the joint loss to train Arjun, where λ_{MT} , λ_{NT} , λ_{MS} , and λ_{CS} are hyper-parameters that control the contribution of each objective to the final loss.

$$L_{\text{joint}}(\theta, x) = \lambda_{MT} L_{MT}(\theta, x) + \lambda_{NT} L_{NT}(\theta, x) + \lambda_{MS} L_{MS}(\theta, x) + \lambda_{CS} L_{CS}(\theta, x) \quad (5)$$

We set all objective weights equally ($\lambda_{MT} = \lambda_{MS} = \lambda_{CS} = \lambda_{NT} = 1.0$) to balance gradient contributions and prevent optimization bias toward any single task. This ensures that the model learns representations that are both locally granular and globally coherent. Empirical validation of this design choice is detailed in Appendix D.

Our goal is to find the optimal model parameters θ^* that minimize the expected joint loss over the corpus D :

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{x \sim D} [L_{\text{joint}}(\theta, x)] \quad (6)$$

This integrated approach ensures the model learns to develop robust representations that capture both bidirectional context (primarily from encoder-focused tasks) and generative capabilities (from decoder-focused tasks), while efficiently sharing parameters across all training signals.

²Note that while English is used here for clarity of demonstration, our actual implementation uses the respective scripts of Bangla, Tamil, and Hindi languages.

4 Experimental Setup

4.1 Data

We conducted pre-training experiments using the Vācaspati dataset (Bhattacharyya et al., 2023) (11M sentences, 115M words) to train **Vac-Arjun**. Additionally, we trained monolingual variants: **Indic-BangArjun**, **Indic-TamArjun**, and **Indic-HindArjun**, on their respective IndicCorp v2 (Kakwani et al., 2020) subsets: Bangla (19M sentences, 898M words), Tamil (12M sentences, 483M words), and Hindi (105M sentences, 6.1B words). All variants utilize identical model architectures and hyper-parameters.

Table 2 summarizes the 20 dataset configurations used for evaluating our models across three languages. The evaluation spans literary, social media, and generative domains, and complex reasoning datasets including Natural Language Inference (NLI) and Question Answering (QA). The collection represents a significant variety in size and difficulty, ensuring a rigorous test of the model’s versatility.

4.2 Pre-training Implementation Details

Arjun employs a Transformer encoder-decoder architecture (Vaswani et al., 2017) enhanced with Disentangled Attention (He et al., 2020), Rotary Position Embeddings (RoPE) (Su et al., 2021), and GeGLU activations (Warner et al., 2024). We implemented three model scales: **Small** (17M total parameters; 3 layers per stack, 256 hidden dimension, 1024 FFN, 4 heads), **Base** (69M parameters; 6 layers per stack, 512 hidden dimension, 2048 FFN, 8 heads), and **Large** (256M parameters; 12 layers per stack, 768 hidden dimension, 3072 FFN, 16 heads).

All configurations process a maximum sequence length of 256 tokens, apply a 0.1 dropout rate, and share weights between the embedding and pre-softmax linear layers (Press and Wolf, 2017). We trained language-specific WordPiece tokenizers (Devlin et al., 2018) with a 50,000-token vocabulary. Models were optimized using AdamW with an initial learning rate of 5×10^{-5} and linear decay. We adopted a lower starting learning rate to ensure stability; Appendix B demonstrates that higher rates induce catastrophic gradient divergence in our multi-objective regime. Training ran for 20 epochs with a batch size of 256 (yielding $\approx 900K$ steps for the Vācaspati corpus) distributed across an AWS p4de.24xlarge instance (8×NVIDIA A100 40GB

Task	Encoder Input	Encoder Output	Decoder Input	Decoder Output
MT	[MT] Arjun model [MASK] optimized for [MASK] tasks	[-100] [-100] [-100] is [-100] [-100] multiple [-100]	-	-
MS	[MS] Arjun model [SPAN1] for multiple [SPAN2]	-	[SOS] [SPAN1] is optimized [SPAN2] tasks	[SPAN1] is optimized [SPAN2] tasks [EOS]
NT	-	-	[NT] Arjun model is optimized for multiple tasks	Arjun model is optimized for multiple tasks [EOS]
CS	[CS] Arjun model	-	[SOS] is optimized for multiple tasks	is optimized for multiple tasks [EOS]

Table 1: Examples of encoder-decoder inputs and outputs for our four pre-training tasks using a sample sentence “Arjun model is optimized for multiple tasks”. For MT, encoder outputs show token indices where [-100] indicates masked positions not used in loss computation.

Lang	Dataset (Task)	Train	Val	Test	Description & Source
Bangla	Poem Classif. (5-cls Classif.)	1.2K	-	299	Tagore’s poems (devotion, love, nature, etc.) (Bhattacharyya et al., 2023)
	Sent. 2-class (Bin. Classif.)	8.6K	-	2.1K	YouTube drama reviews (Pos/Neg) (Sazzed, 2020)
	Sent. 3-class (3-cls Classif.)	12.1K	-	1.5K	Social media comments (Pos/Neg/Neu) (Islam et al., 2021)
	WikiANN (NER)	20.2K	2.9K	2.6K	Wikipedia NER (PER, LOC, ORG, O) (Pan et al., 2017)
	Spelling Error (Bin. Classif.)	15.2K	-	3.8K	Context-sensitive spelling error detection (Bhattacharyya et al., 2023)
	Paraphrasing (Generation)	419K	-	23K	Sentence-level paraphrasing (Akil et al., 2022)
	GEC (Generation)	535K	-	50K	Grammatical Error Correction (Bhattacharyya and Bhattacharya, 2025b)
	IndicXNLI (NLI)	393K	2.5K	5K	Natural Language Inference translated from XNLI (Aggarwal et al., 2022)
Tamil	Bangla QA (QA)	118K	2.5K	2.5K	Diverse QA pairs (20% sampled test) (Bhattacharjee et al., 2022)
	News Category (3-cls Classif.)	5.3K	669	669	News topic prediction (Kakwani et al., 2020)
	WikiANN (NER)	20.4K	2.5K	2.6K	Wikipedia NER subset (Pan et al., 2017)
	IndicXNLI (NLI)	393K	2.5K	5K	Natural Language Inference translated from XNLI (Aggarwal et al., 2022)
Hindi	IndicQA (QA)	1.5K	-	300	Extractive reading comprehension (Doddapaneni et al., 2022)
	BBC News (14-cls Classif.)	3.4K	-	866	Fine-grained news topic classification (Kakwani et al., 2020)
	WikiANN (NER)	9.4K	1.1K	1.2K	Wikipedia NER subset (Pan et al., 2017)
	Movie Review (3-cls Classif.)	2.4K	310	310	Combined sentiment analysis (IIT-P, IIT-B, iNLTK) (Kakwani et al., 2020)
	Product Review (3-cls Classif.)	4.1K	523	523	Product review sentiment (Pos/Neg/Neu) (Kakwani et al., 2020)
	MIDAS Disc. (6-cls Classif.)	7.9K	997	997	Discourse mode classification (Dhanwal et al., 2020)
	IndicXNLI (NLI)	392K	3.2K	5K	Natural Language Inference translated from XNLI (Aggarwal et al., 2022)
XQuAD (QA)		1K	-	190	Multilingual QA translated from SQuAD v1.1 (Artetxe et al., 2019)

Table 2: Summary of the 20 downstream dataset configurations for model evaluation in Bangla, Tamil, and Hindi.

GPUs), completing in 7–20 days depending on the corpus size.

4.3 Baselines and Fine-tuning

Discriminative tasks utilized only the encoder (Arjun-Base: 44M, Arjun-Large: 143M), while generative tasks employed the full architecture (Arjun-Base: 69M, Arjun-Large: 256M). All models were fine-tuned via AdamW (constant learning rate 5×10^{-6}). Discriminative tasks were evaluated using Macro-F1. For generative tasks, Paraphrasing was evaluated using sacreBLEU, while Grammatical Error Correction (GEC) was evaluated using MaxMatch ($M^2 F_{0.5}$) and GLEU to strictly penalize trivial copying and reward precise edits. To manage our multi-objective routing during downstream fine-tuning, we retain the prompt token most aligned with the target task’s structure: we prepend the [MT] token for all discriminative tasks and the [CS] token for generative tasks. We empirically validated this mapping, confirming that aligning the downstream prompt with the closest pre-training objective yields optimal convergence. To isolate architectural impact, all baselines were

pre-trained from scratch on our exact matched corpora using identical hyperparameters (batch 256, 900K steps).

Bangla Discriminative Baselines. To evaluate *Vac-Arjun* and *Indic-BangArjun* across scales (Small: 17M, Base: 44M, Large: 143M), we pre-trained encoder-only baselines on the Vācaspati and IndicCorp-v2 corpora: *ModernBERT* (151M) (Warner et al., 2024), *DeBERTaV3* (138M) (He et al., 2021), *IndicBERT-v2-MLM* (278M) (Doddapaneni et al., 2022), and a parameter-matched *UL2* (Tay et al., 2022). We also included *Vac-BERT* (17M) (Bhattacharyya et al., 2023) to benchmark Arjun-Small at the sub-20M scale.

Bangla Generative Baselines. On the combined (Vācaspati + IndicCorp v2) corpus, we benchmarked Arjun (Base: 69M, Large: 256M) against *BanglaT5* (247M) (Bhattacharjee et al., 2023), *BanglaByT5* (330M) (Bhattacharyya and Bhattacharya, 2025a), a matched *UL2* (69M), and two decoder-only *GPT-2* models (69M, 355M) (Radford et al., 2019) to isolate architectural advantages.

Model	Params	Sentiment		Poem	NER	Spell	NLI	QA
		2-cls	3-cls					
Pre-trained on Vācaspati Corpus								
IndicBERT-v2	278M	94.06	68.55	61.09	92.02	81.07	69.05	72.15/78.88
ModernBERT	151M	94.33	68.67	61.98	92.05	81.13	69.15	74.03/82.37
DeBERTaV3	138M	94.00	68.43	61.23	92.00	81.11	68.53	73.88/81.87
UL2	44M	93.53	66.78	59.14	91.05	78.87	69.14	73.16/81.09
VAC-BERT	17M	94.75	68.79	60.39	89.70	77.72	68.31	69.93/78.03
Arjun-Small	17M	95.42	69.31	62.91	91.55	80.71	69.45	74.12/81.01
Arjun-Base	44M	95.81	70.78	65.94	94.12	83.15	72.31	76.51/83.72
Arjun-Large	143M	96.10	71.45	67.20	95.05	83.80	74.50	77.85/85.10
Pre-trained on IndicCorp v2 Bangla Corpus								
IndicBERT-v2	278M	93.93	68.61	60.91	91.65	80.73	69.82	73.98/80.23
ModernBERT	151M	94.13	68.58	62.08	92.34	81.03	71.33	74.89/83.05
DeBERTaV3	138M	94.06	68.17	61.88	91.89	80.69	70.99	74.05/82.56
UL2	44M	93.21	67.19	61.28	91.87	77.98	70.04	74.85/82.77
Indic-BangArjun-Base	44M	96.02	71.95	68.12	95.21	82.44	75.02	77.21/85.12
Indic-BangArjun-Large	143M	96.23	72.02	68.98	96.01	82.93	75.57	77.98/86.03

Table 3: Controlled performance comparison (Macro-F1 / F1/EM for QA) on Bangla **discriminative** tasks.

Hindi and Tamil Baselines. For discriminative tasks in Hindi and Tamil, we pre-trained ModernBERT, DeBERTaV3, IndicBERT-v2-MLM, and UL2 from scratch on the respective IndicCorp v2 subsets, evaluating against Arjun Base and Large.

5 Results

We present a comprehensive evaluation of our multi-objective pre-training framework across three Indic languages: Bangla, Tamil, and Hindi. We benchmark our models against state-of-the-art language-specific models, assessing efficacy across a diverse range of downstream applications spanning discriminative and generative paradigms. To ensure experimental reliability, all reported results are averages of 10 independent runs. **Bold** indicates results that are better with statistical significance ($p < 0.005$) compared to the runner-up. Extended result tables detailing standard deviations, and additional tasks, are provided in Appendix C.

5.1 Bangla Supervised Tasks

Table 3 and Table 4 present the results of our controlled experiments on Bangla downstream tasks. By strictly matching the pre-training corpora, we isolate the architectural contributions of Arjun from data-driven gains.

Discriminative Task Evaluation (Vācaspati and IndicCorp v2). When pre-trained on the Vācaspati corpus, Arjun-Base (44M) consistently outperforms the parameter-matched UL2 baseline across all tasks, validating the superior efficiency of our multi-objective formulation. Notably, on the complex 3-class Sentiment Analysis and Spell Error Detection tasks, Arjun-Base surpasses significantly larger models, including ModernBERT (151M) and DeBERTaV3 (138M). On complex

NLU evaluation categories like Natural Language Inference (IndicXNLI) and Question Answering (BanglaQA), Arjun-Base establishes a clear performance margin, achieving a 72.31 Macro-F1 score on NLI and a 76.51/83.72 F1/EM on QA.

Furthermore, we observe that our improvements scale down effectively. The Arjun-Small (17M) model outperforms the existing VAC-BERT (17M) on every benchmark, yielding a 1.85 percentage point (pp) gain in NER and a 2.99 pp gain in Spell Error detection, confirming that our architecture provides better parameter utilization at the sub-20M scale. Our scaled-up variant, Arjun-Large (143M), achieves the highest performance among all evaluated models within matched parameter budgets across all downstream evaluations.

The trend holds when scaling to the larger IndicCorp v2 dataset. Indic-BangArjun-Base (44M) demonstrates exceptional architectural efficiency among the compared models, outperforming the substantially larger monolingual IndicBERT-v2 (MLM-only, 278M) on all tasks. On QA and NLI tasks, Indic-BangArjun-Base yields prominent gains over ModernBERT (+3.69 pp in NLI). Finally, Indic-BangArjun-Large (143M) extends these gains further, validating that the benefits of multi-objective pre-training scale symmetrically with model parameters.

Generative Task Evaluation (Combined Corpus). To evaluate generative capabilities, we pre-train models on the combined Bangla dataset (Table 4). In previous iterations, sentence-rewriting tasks like GEC evaluated via sacreBLEU exhibited extreme metric artifacts due to high source-target token overlap. Baselines like BanglaByT5 and UL2 suffered from "over-generation"—unnecessarily paraphrasing the source—which heavily penalized standard n-gram matching, whereas Arjun exhibited highly conservative editing. To ensure rigorous, scientifically plausible evaluation, we report GEC using MaxMatch ($M^2F_{0.5}$) and GLEU. Under these strict metrics, Arjun-Base (69M) demonstrates exceptional parameter efficiency. It significantly outperforms both parameter-matched UL2 and 3.5–4.8 $\times$ larger encoder-decoder baselines (BanglaByT5 and BanglaT5), achieving an $M^2F_{0.5}$ of 54.3 compared to BanglaByT5’s 48.7. To further validate our architectural choices, we benchmarked decoder-only GPT-2 configurations; Arjun significantly outperforms these architectures, highlighting the necessity of bidirectional contextual

Model	Parameters	Para. sacreBLEU	GEC	
			$M^2F_{0.5}$	GLEU
<i>Pre-trained on Vācaspati + IndicCorp v2 (Combined)</i>				
BanglaByT5	330M	35.78	48.7	55.4
BanglaT5	247M	34.97	47.1	53.9
UL2	69M	37.22	45.2	52.8
GPT-2	69M	22.89	29.4	34.1
GPT-2	355M	27.16	33.7	38.6
Arjun-Base	69M	40.45	54.3	61.2
Arjun-Large	256M	42.15	55.9	62.8

Table 4: Performance comparison on Bangla **generative** tasks. GEC is evaluated using MaxMatch ($M^2F_{0.5}$) and GLEU to account for source-target overlap.

Model	Params	BBC News	NER	IIT Patna		MIDAS Disc.	NLI	QA (XQuAD)
				Mov	Prod			
IndicBERT-v2	278M	55.71	88.43	68.41	76.28	66.04	72.94	46.03/62.89
UL2	44M	60.05	89.67	69.91	78.94	67.25	74.38	47.12/65.22
ModernBERT	151M	60.08	89.92	69.88	79.00	67.72	74.23	47.73/65.63
DeBERTaV3	138M	59.16	88.86	69.57	77.45	67.80	74.29	47.87/65.15
HindArjun-Base	44M	62.68	91.95	71.72	79.58	70.04	77.34	49.81/67.75
HindArjun-Large	143M	63.92	93.12	73.04	81.25	71.88	79.45	51.62/69.84

Table 5: Performance comparison (Macro-F1 / F1/EM for QA) on Hindi discriminative tasks.

anchors for precise rewriting tasks. The introduction of Arjun-Large (256M) establishes a highly competitive baseline for parameter-efficient Bangla generation. To validate these automated metrics against data contamination and qualitative alignment, a comprehensive Data Leakage Audit and Human Evaluation are detailed in Appendix G.

5.2 Tamil and Hindi Supervised Tasks

We extended our controlled evaluation to Tamil and Hindi, pre-training models specifically on the respective IndicCorp v2 subsets. Table 5 and Table 6 summarize the performance of Arjun against parameter-matched (UL2) and larger (ModernBERT, DeBERTaV3, IndicBERT-v2) baselines.

In Hindi, Indic-HindArjun-Base (44M) demonstrates remarkable efficiency, achieving the highest scores across all seven tasks compared to the baselines. It notably outperforms the 151M parameter ModernBERT on BBC News Classifica-

Model	Params	News Class.	NER	NLI	QA (IndicQA)
IndicBERT-v2	278M	96.70	83.77	72.28	30.78 / 45.09
UL2	44M	96.69	86.97	73.00	31.64 / 44.56
ModernBERT	151M	97.56	87.21	73.08	31.77 / 44.99
DeBERTaV3	138M	96.99	87.09	72.87	31.52 / 44.68
TamArjun-Base	44M	98.65	90.62	74.02	33.12 / 47.52
TamArjun-Large	143M	99.21	92.15	76.48	35.82 / 50.14

Table 6: Performance comparison (Macro-F1 / F1/EM for QA) on Tamil discriminative tasks.

tion (62.68% vs 60.08%) and structurally demanding tasks like IndicXNLI (+3.11 percentage points over ModernBERT). Similarly, for Tamil, Indic-TamArjun-Base maximizes performance for its parameter class, surpassing the much larger ModernBERT by 3.41 percentage points in NER and outperforming all baselines on the IndicQA benchmark.

Across both languages, the addition of the 143M parameter variant, Arjun-Large, systematically amplifies these architectural advantages. Indic-HindArjun-Large outperforms the comparable IndicBERT-v2 baseline by a wide margin (+6.51 in Hindi NLI and +5.59 percentage points respectively in XQuAD F1 score), demonstrating that computing losses directly at the encoder-side within a unified framework generates superior cross-functional representations.

5.3 Performance in the Era of Large Language Models

While massive multilingual LLMs achieve strong zero-shot performance, their deployment in low-latency or edge environments remains computationally prohibitive. To contextualize Arjun’s utility, we benchmarked Arjun-Base against modern open-weight LLMs (evaluated 5-shot) on Bangla tasks (Table 7).

Importantly, this is not an apples-to-apples comparison: LLMs are pre-trained on trillions of tokens, whereas Arjun uses a constrained, language-specific corpus. Yet, Arjun’s bidirectional encoder dominance allows it to drastically outperform the 32B model across 5 out of 7 discriminative tasks, enforcing strict structural compliance (e.g., exact NER boundaries) where causal LLMs struggle. On generative tasks, the massive world-knowledge of LLMs yields a slight advantage, but at an extreme computational premium. Arjun-Base requires $< 1\text{GB}$ VRAM, delivering $21\times$ faster latency and $48\times$ more throughput than a 32B model, validating its necessity for resource-constrained deployment.

5.4 Architectural Analysis: Multi-Objective Synergy

Despite identical parameter counts and data, Arjun outperforms UL2 due to distinct loss application. While UL2 unifies objectives via data transformation (R-, S-, and X-denoisers), it fundamentally optimizes a single sequence-to-sequence loss at the decoder (Tay et al., 2022). Similarly, frameworks

Model	Params	Sent 2	Sent 3	Poem	NER	Spell	NLI	QA	Para	GEC (M^2)	Latency (ms)*	Throughput*
Qwen2.5-32B	~32B	93.1	69.4	68.5	93.4	78.5	75.7	76.1	42.8	56.4	320.5	45
Llama-3.1-8B	~8B	90.2	67.1	67.3	92.8	77.2	73.5	75.8	41.5	55.2	150.2	150
Gemma-2-2B	~2B	90.5	66.4	67.1	92.3	75.4	74.1	75.4	40.8	55.5	85.4	350
Arjun-Base	69M	96.0	71.9	68.1	95.2	82.4	74.3	76.5	40.4	54.3	14.8	2,160

Table 7: Benchmarking Arjun-Base against modern open-weight LLMs (5-shot) on Bangla tasks. *Inference profiled on a single NVIDIA A10G GPU (Batch Size = 32, ms/seq and seq/sec).

like SpacTor-T5 (Ye et al., 2024), GLM (Du et al., 2022), and UniLM collapse pre-training objectives into unified decoder losses or singular masking mechanisms, leaving the encoder to receive gradient updates only indirectly.

Conversely, Arjun employs a “multi-front” optimization: we compute the MT loss explicitly at the encoder layer independently of the decoder, while simultaneously calculating NT and MS/CS generation losses at the decoder. This direct supervision forces the encoder to learn robust, standalone discriminative representations—crucial for NER and NLI—while the decoder-focused objectives ensure generative competence without compromising the encoder’s bidirectional integrity.

This advantage extends to single-objective baselines. Models like ModernBERT and DeBERTa (MLM/RTD) lack the causal reasoning fostered by autoregressive training, while T5-variants (e.g., BanglaT5) rely on span corruption, limiting fine-grained token-level awareness. Arjun bridges these paradigms by jointly optimizing MT, NT, MS, and CS. Figure 2 validates this by displaying the per-objective loss trajectories during multi-objective Arjun training; their stable, parallel convergence without divergence confirms that our objectives operate in concert rather than in competition.

5.5 Ablation Studies

To evaluate the synergistic contributions of our pre-training objectives, we conducted an exhaustive combinatorial ablation over all 15 task configurations (Table 8). Using the Bangla Vācaspati corpus, we tracked performance across both discriminative and generative paradigms. Because the Next Token (NT) objective operates strictly on the decoder, the NT-only configuration cannot yield encoder representations for discriminative tasks.

The progression confirms that cumulative objective density tightly correlates with natural downstream gains. Single-task generative baselines (MS/CS) yield outputs comparable to BanglaT5, while the standalone NT objective aligns with

Configuration	Poem	NER	NLI	QA	Para	GEC	Time(s)
MT	61.35	92.12	69.10	73.85	—	—	0.50
MS	61.12	92.20	68.95	73.50	34.70	70.15	0.67
CS	61.40	92.35	69.18	74.10	35.40	71.68	0.67
NT	—	—	—	—	24.15	63.02	0.52
MT + MS	64.25	92.50	70.15	74.30	36.50	80.42	0.88
MT + CS	64.38	92.62	70.42	74.65	36.10	81.10	0.88
MT + NT	62.10	91.80	69.35	73.92	28.80	72.85	0.85
MS + CS	63.95	92.48	70.12	74.22	37.20	85.55	0.95
MS + NT	61.85	91.15	68.70	73.08	36.25	84.12	0.90
CS + NT	62.04	91.32	68.94	73.40	36.90	86.40	0.92
MT + MS + CS	64.92	92.95	71.50	75.12	38.10	94.10	1.20
MT + MS + NT	64.50	92.71	70.85	74.80	38.40	94.95	1.18
MT + CS + NT	64.62	92.80	71.02	75.01	38.20	94.62	1.19
MS + CS + NT	64.15	92.55	70.38	74.45	39.85	95.10	1.22
Arjun-Base (Full)	65.94	94.12	72.31	75.93	40.45	97.12	1.38

Table 8: Combinatorial ablation results (Macro-F1 for classification/NLI, F1 for QA, sacreBLEU for Para/GEC) and training time per step on Bangla tasks.

decoder-only GPT-2 architectures. The full Arjun-Base framework establishes the highest scores across all modalities. Crucially, the NT objective’s impact depends heavily on topology:

(1) Generative Acceleration: Incorporating NT yields uniform gains in all 12 generative comparisons.

(2) Lower-Order Interference: When adding NT to single-task bidirectional anchors (MT, MS, or CS), its unconstrained sequential bias interferes with bidirectional extraction. Out of 45 lower-order discriminative metric comparisons across Bangla, Hindi, and Tamil, adding NT improved performance in 35 instances but caused direct regressions in 10 (primarily degrading NER and NLI).

(3) Higher-Order Synergy: This interference resolves entirely in complex architectures. Transitioning from the 3-task baseline (MT+MS+CS) to the full 4-task framework provides robust bidirectional anchors that neutralize sequential drift, allowing NT to act as a global multiplier. This transition improves all 17 evaluated metrics across all three languages. Cross-lingual ablations tracking these trends are in Appendix E.

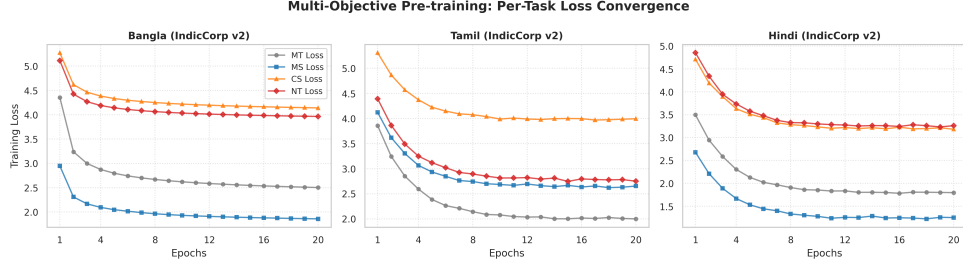


Figure 2: Per-objective training loss trajectories for Arjun-Base across IndicCorp v2 corpora.

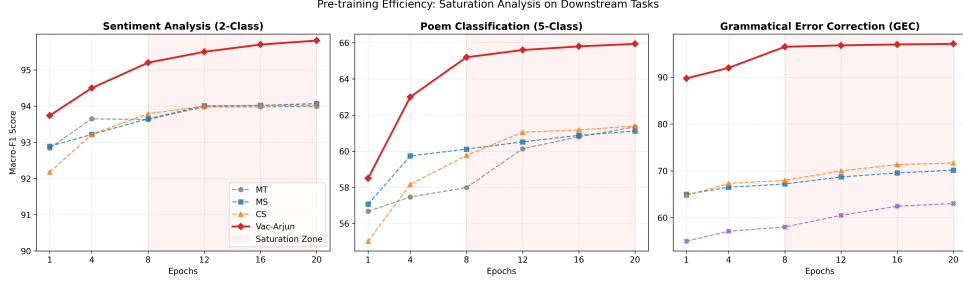


Figure 3: Performance saturation analysis across training epochs. Vac-Arjun (red solid line) demonstrates faster convergence and higher ceilings compared to single-task baselines (dashed lines).

5.6 Pre-training Efficiency and Saturation

To understand the efficiency of the multi-task pre-training approach, we analyzed the downstream performance of intermediate checkpoints saved every 4 epochs. Figure 3 illustrates the performance trajectory of Vac-Arjun compared to single-task baselines (MT, MS, CS) across Sentiment Analysis, Poem Classification, and Grammatical Error Correction (GEC).

The results demonstrate that Vac-Arjun not only achieves superior final performance but also reaches saturation significantly faster. In the Poem Classification task, Vac-Arjun converges to near-peak performance (Macro-F1 $\approx 65\%$) by Epoch 8, whereas the single-task baselines plateau around 60-61% even after 20 epochs. A similar trend is observed in Sentiment Analysis, where the full model establishes a clear lead early in training. For the generative GEC task, while all models show continuous improvement, Vac-Arjun maintains a consistent performance gap ($\approx 4-5$ points) over the best single-task baseline throughout the training process. This suggests that the synergistic effect of the four objectives accelerates the learning of robust linguistic features, allowing the model to become downstream-ready with fewer pre-training steps compared to single-objective counterparts.

6 Conclusions and Future Directions

We presented a novel parameter-efficient pre-training framework combining four complementary objectives. With just 69M parameters, our Arjun models match or outperform $2.5-4\times$ larger models across 20 Bangla, Tamil, and Hindi tasks. Ablation studies confirm that jointly optimizing all four objectives yields synergistic gains over single-task baselines, validating the fusion of bidirectional and autoregressive modeling. Future work will explore integrating replaced token detection (ELECTRA) and extending this multi-objective framework to broader monolingual and multilingual contexts.

Furthermore, evaluating the decoder’s long-term discourse coherence on open-ended generation and document summarization is a critical next step for our upcoming multilingual scaling efforts.

Acknowledgements

Following ARR’s AI Writing/Coding Assistance Policy, we acknowledge using Claude 3.7 Sonnet (Anthropic, 2025) for editorial assistance while maintaining full ownership of all scientific contributions.

7 Limitations

While our multi-objective pre-training approach demonstrates promising results for parameter-efficient Indic language models, several limitations

should be acknowledged. First, our evaluation focuses on only three Indian languages (Bangla, Tamil, and Hindi) out of the hundreds spoken in the Indian subcontinent. While we validated Arjun’s performance on the largest available Hindi corpus (105M sentences, 6.1B words), its scalability to even larger, web-scale corpora remains to be tested; furthermore, the effectiveness of our approach on highly agglutinative or tonal languages requires further empirical validation.

Second, although we have expanded our evaluation to include generative tasks such as Paraphrasing and Grammatical Error Correction, these are primarily sentence-level rewriting tasks. We have not yet extensively explored the model’s capabilities in long-form text generation, open-ended creative writing, or tasks requiring complex, multi-step reasoning. Consequently, the full potential of the decoder component for maintaining discourse coherence over long contexts remains an area for future investigation.

Third, while we trained separate monolingual models to isolate language-specific gains, we have not thoroughly investigated the cross-lingual transfer capabilities of our architecture. The potential for knowledge sharing across related Indic languages through multilingual joint training remains an open avenue. Furthermore, our reliance on existing benchmark datasets, some of which are relatively small (e.g., poem classification), may limit the granularity of our analysis regarding specific linguistic phenomena.

Fourth, we acknowledge that empirical macro-level synergy does not entirely preclude underlying representational conflict. While our joint loss converges smoothly, this optimization might reflect a calculated compromise across the parameter space rather than purely frictionless synergy. The exact degree of gradient interference or task-specific parameter "cannibalization" between bidirectional and autoregressive objectives remains an open question. We believe this represents a foundational constraint in current multi-objective pre-training, and we suggest that future work utilize targeted probing studies to disentangle these internal dynamics.

Fifth, regarding our multi-objective regime, we observe that the joint optimization of bidirectional and autoregressive losses creates a highly non-convex loss landscape. As detailed in Appendix B, our framework is highly sensitive to optimization hyperparameters; elevated learning rates that succeed in single-objective training induce catas-

trophic gradient divergence when optimizing all four tasks simultaneously. Future work should investigate gradient-balancing techniques to stabilize this dynamic. Furthermore, while we thoroughly evaluated generative competence via Paraphrasing and GEC, these are strictly sentence-rewriting tasks. Testing the model’s capabilities on long-form, open-ended generation remains a critical next step to fully validate the decoder’s autoregressive coherence.

Finally, the primary scope of our work is parameter-efficient fine-tuning for downstream NLU/NLG tasks. Evaluating our framework against massively multilingual bitext models (e.g., LASER3, LaBSE) specifically for cross-lingual sentence retrieval remains an area for future evaluation.

8 Ethics Statement

The development and deployment of language models for Indian languages raises several important ethical considerations. Our models are trained on text corpora that may not equally represent all speakers of Bangla, Tamil, and Hindi. The Vācaspati corpus focuses on literary texts, while IndicCorp v2 primarily contains web-scraped content, potentially underrepresenting certain demographic groups, dialects, or socioeconomic backgrounds. We acknowledge that model performance may vary across different user groups, and we encourage future work to evaluate fairness across diverse populations within each language community.

Working with Indian languages requires careful consideration of cultural nuances, historical context, and community values. We recognize that our computational approach may not capture the full cultural richness embedded in these languages. We have made efforts to use established, community-validated datasets and evaluation benchmarks where possible, but acknowledge that automated systems may inadvertently perpetuate or amplify existing societal biases present in training data.

Training multiple language-specific models requires significant computational resources, contributing to carbon emissions and environmental impact. While our parameter-efficient approach reduces these costs compared to larger models, we acknowledge the environmental responsibility associated with computational research. We recommend that future applications of this work consider

the environmental trade-offs and explore carbon offset options where possible.

The training corpora used in this work were collected from various sources, including web-scraped content. While we relied on established, publicly available datasets, we recognize the importance of data consent and privacy. Future work should prioritize the development of ethically sourced datasets with explicit consent from content creators and communities. Language models can potentially be misused for generating harmful content, spreading misinformation, or automating deceptive practices. While our evaluation focuses on beneficial applications, we acknowledge that our models could theoretically be applied to harmful uses. We encourage responsible deployment and recommend implementing appropriate safeguards and use-case restrictions in practical applications.

While our work aims to democratize NLP technology for Indian languages, the computational requirements may still create barriers for researchers and practitioners in resource-constrained environments. We are committed to making our models and findings publicly available to promote equitable access to this technology. We recognize that meaningful progress in Indian language technology requires ongoing engagement with native speaker communities, linguists, and cultural experts. Our current work represents a computational contribution, but we acknowledge the importance of incorporating community feedback and ensuring that technological advances align with community needs and values. We commit to providing detailed documentation of our methods, hyperparameters, and evaluation procedures to ensure reproducibility and plan to release our trained models and code to enable further research and community validation of our findings.

References

- Divyanshu Aggarwal, Vivek Gupta, and Anoop Kunchukuttan. 2022. [IndicXNLI: Evaluating multilingual inference for Indian languages](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10994–11006, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Ajwad Akil, Najrin Sultana, Abhik Bhattacharjee, and Rifat Shahriyar. 2022. [BanglaParaphrase: A high-quality Bangla paraphrase dataset](#). In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 261–272. Association for Computational Linguistics.
- Anthropic. 2025. Claude 3.7 sonnet [large language model]. <https://www.anthropic.com>. Released February 24, 2025. Accessed 2025-05-19.
- Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2019. [On the Cross-lingual Transferability of Monolingual Representations](#). *arXiv e-prints*, arXiv:1910.11856.
- Abhik Bhattacharjee, Tahmid Hasan, Wasi Ahmad, Kazi Samin Mubasshir, Md Saiful Islam, Anindya Iqbal, M. Sohel Rahman, and Rifat Shahriyar. 2022. [BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla](#). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1318–1327, Seattle, United States. Association for Computational Linguistics.
- Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, and Rifat Shahriyar. 2023. [BanglaNLG and BanglaT5: Benchmarks and resources for evaluating low-resource natural language generation in Bangla](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 726–735, Dubrovnik, Croatia. Association for Computational Linguistics.
- Pramit Bhattacharyya and Arnab Bhattacharya. 2025a. [BanglaByT5: Byte-level modelling for Bangla](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5551–5560, Suzhou, China. Association for Computational Linguistics.
- Pramit Bhattacharyya and Arnab Bhattacharya. 2025b. [Leveraging LLMs for Bangla grammar error correction: Error categorization, synthetic data, and model evaluation](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8220–8239, Vienna, Austria. Association for Computational Linguistics.
- Pramit Bhattacharyya, Joydeep Mondal, Subhadip Maji, and Arnab Bhattacharya. 2023. [VACASPATI: A diverse corpus of Bangla literature](#). In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1130, Nusa Dua, Bali. Association for Computational Linguistics.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators](#). *arXiv e-prints*, arXiv:2003.10555.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised](#)

- cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). *arXiv e-prints*, arXiv:1810.04805.
- Swapnil Dhanwal, Hritwik Dutta, Hitesh Nankani, Nilay Shrivastava, Yaman Kumar, Junyi Jessy Li, Debanjan Mahata, Rakesh Gosangi, Haimin Zhang, Rajiv Ratn Shah, and Amanda Stent. 2020. [An annotated dataset of discourse modes in Hindi stories](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1191–1196, Marseille, France. European Language Resources Association.
- Sumanth Doddapaneni, Rahul Aralikkatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2022. [Towards Leaving No Indic Language Behind: Building Monolingual Corpora, Benchmark and Models for Indic Languages](#). *arXiv e-prints*, arXiv:2212.05409.
- Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. [GLM: General language model pretraining with autoregressive blank infilling](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, Dublin, Ireland. Association for Computational Linguistics.
- Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. [DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing](#). *arXiv e-prints*, arXiv:2111.09543.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. [DeBERTa: Decoding-enhanced BERT with Disentangled Attention](#). *arXiv e-prints*, arXiv:2006.03654.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. [LoRA: Low-Rank Adaptation of Large Language Models](#). *arXiv e-prints*, arXiv:2106.09685.
- Khondoker Ittehadul Islam, Sudipta Kar, Md Saiful Islam, and Mohammad Ruhul Amin. 2021. [SentNoB: A dataset for analysing sentiment on noisy Bangla texts](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3265–3271, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. [SpanBERT: Improving Pre-training by Representing and Predicting Spans](#). *arXiv e-prints*, arXiv:1907.10529.
- Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. [IndicNLP Suite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.
- Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, Shruti Gupta, Subhash Chandra Bose Gali, Vish Subramanian, and Partha Talukdar. 2021. [MuRIL: Multilingual Representations for Indian Languages](#). *arXiv e-prints*, arXiv:2103.10730.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. [ALBERT: A Lite BERT for Self-supervised Learning of Language Representations](#). *arXiv e-prints*, arXiv:1909.11942.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#). *arXiv e-prints*, arXiv:1907.11692.
- Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. [Cross-lingual name tagging and linking for 282 languages](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.
- Ofir Press and Lior Wolf. 2017. [Using the output embedding to improve language models](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. [Improving language understanding by generative pre-training](#). *OpenAI*.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#). *arXiv e-prints*, arXiv:1910.10683.
- Salim Sazzed. 2020. [Cross-lingual sentiment classification in low-resource Bengali language](#). In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 50–60, Online. Association for Computational Linguistics.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. [RoFormer: Enhanced Transformer with Rotary Position Embedding](#). *arXiv e-prints*, arXiv:2104.09864.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2022. [UL2: Unifying Language Learning Paradigms](#). *arXiv e-prints*, arXiv:2205.05131.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention Is All You Need](#). *arXiv e-prints*, arXiv:1706.03762.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. [Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference](#). *arXiv e-prints*, arXiv:2412.13663.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjape, Adina Williams, Tal Linzen, and Ryan Cotterell. 2025. [Findings of the BabyLM Challenge: Sample-Efficient Pretraining on Developmentally Plausible Corpora](#). *arXiv e-prints*, arXiv:2504.08165.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. [mT5: A massively multilingual pre-trained text-to-text transformer](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [XLNet: Generalized Autoregressive Pretraining for Language Understanding](#). *arXiv e-prints*, arXiv:1906.08237.

Ke Ye, Heinrich Jiang, Afshin Rostamizadeh, Ayan Chakrabarti, Giulia DeSalvo, Jean-François Kagy, Lazaros Karydas, Gui Citovsky, and Sanjiv Kumar. 2024. [SpacTor-T5: Pre-training T5 Models with Span Corruption and Replaced Token Detection](#). *arXiv e-prints*, arXiv:2401.13160.

Appendix

A Representation Balance and Objective Synergy

The empirical results demonstrate that deviating from an equal weighting scheme generally degrades overall performance, particularly hurting the

model’s versatility. We attribute this behavior to the shared parameter space within the transformer encoder-decoder architecture:

- **Modality Collapse:** Over-weighting a single objective forces the shared hidden states to over-optimize for one specific type of linguistic dependency. For instance, the MT-Heavy configuration aggressively penalizes token-level errors, forcing the encoder to hyper-focus on local context. While one might expect this to improve token-level tasks like NER, it actually strips the encoder of the broader, multi-token structural awareness ordinarily provided by the MS and CS objectives, causing a slight drop in NER and a severe collapse in generative GEC capabilities.
- **Discriminative Degradation:** Conversely, over-weighting the sequence-to-sequence tasks (MS-Heavy or CS-Heavy) shifts the internal representations too heavily toward autoregressive decoding strategies. This diminishes the encoder’s ability to form robust, standalone bidirectional representations, leading to noticeable performance drops in discriminative tasks like Sentiment Analysis and NER.
- **Equal Weighting as Regularization:** Setting all $\lambda = 1.0$ acts as a natural regularization mechanism. Because the objectives are structurally distinct but semantically complementary, equal gradients force the network into a Pareto optimal state. The shared encoder must learn representations that are simultaneously granular enough to predict single masked tokens (MT), structurally aware enough to construct multi-word spans (MS), and globally coherent enough to generate sentence completions (CS).

Ultimately, symmetric loss weighting prevents the multi-task optimization from collapsing into a pseudo-single-task regime, ensuring the representations remain fully generalized for both encoder-only and encoder-decoder downstream applications.

B Optimization Dynamics and Learning Rate Selection

During the development of the Arjun pre-training pipeline, we analyzed the optimization dynamics across various learning rate regimes to determine the most stable configuration for joint multi-objective training. While modern single-objective pre-training frameworks often utilize larger learning rates (e.g., 5×10^{-4}) to accelerate convergence,

our empirical observations indicate that joint multi-objective optimization fundamentally alters the loss landscape, imposing stricter constraints on hyperparameter selection.

Table 9 details the training trajectory and downstream impact when comparing our chosen learning rate (5×10^{-5}) against a higher rate (5×10^{-4}) on the Vācaspati corpus.

Configuration	Learning Rate	Training Loss (Epoch 3)	Final NER (Macro-F1)
MT (Single-Task)	5×10^{-5}	2.14	91.98
	5×10^{-4}	1.85	92.20
Arjun (MT+MS+CS)	5×10^{-5}	2.31	93.57
	5×10^{-4}	8.75	14.21*

Table 9: Impact of learning rate on training stability and downstream performance. *Indicates catastrophic gradient divergence where the model failed to learn meaningful representations.

B.1 Loss Landscape Complexity

As demonstrated in Table 9, isolated single-objective ablations (such as MT-only) successfully converge and even show marginal downstream improvements when utilizing a higher learning rate of 5×10^{-4} . However, when scaling to the joint optimization of four tasks (MT, MS, CS, and NT), applying this elevated learning rate consistently leads to optimization instability. At 5×10^{-4} , the joint training loss spikes and stagnates (Loss = 8.75 at Epoch 3), resulting in catastrophic gradient divergence and a complete collapse in downstream task capability (14.21 Macro-F1 on NER, equivalent to random performance).

We attribute this instability to the highly non-convex loss landscape generated by the synergy of distinct modeling signals. The network must simultaneously satisfy the gradients for bidirectional contextual understanding (MT) and multi-token/sequence-level generation (MS, CS) within the same shared parameter space. These competing optimization pressures necessitate a more conservative step size to maintain stable convergence and prevent parameter oscillation.

B.2 Convergence Efficiency

Despite the smaller learning rate of 5×10^{-5} , the Arjun framework does not suffer from under-training. As evidenced by the saturation analysis in Figure 3, the model rapidly approaches peak downstream performance by Epoch 8 and plateaus shortly thereafter. This early and sustained satu-

ration confirms that the conservative learning rate provides sufficient momentum to traverse the multi-objective loss landscape, achieving effective and stable convergence well within the standard 20-epoch training window.

C Extended Experimental Result Tables with Standard Deviations

This appendix provides comprehensive experimental results, extending the summary tables presented in Section 5. Specifically, we report the mean and standard deviation across 10 independent runs for all models. These expanded tables include the newly introduced decoder-only baselines (GPT-2) and our scaled-up Arjun-Large variants across all evaluated tasks. Furthermore, we provide detailed performance metrics for all Natural Language Inference (NLI) and Question Answering (QA) datasets, ensuring full transparency of model variance, stability, and scaling characteristics across discriminative workloads.

Model	Parameters	Paraphrasing	GEC
<i>Pre-trained on Vācaspati + IndicCorp v2 (Combined)</i>			
BanglaByT5	330M	35.78 ± 1.60	72.36 ± 1.37
BanglaT5	247M	34.97 ± 1.36	69.13 ± 1.32
UL2	69M	37.22 ± 1.37	73.16 ± 1.29
GPT-2	69M	22.89 ± 1.74	61.91 ± 1.82
GPT-2	355M	27.16 ± 1.55	67.82 ± 1.61
Arjun-Base (Ours)	69M	40.45 ± 1.22	97.12 ± 1.05
Arjun-Large (Ours)	256M	42.15 ± 1.10	97.85 ± 0.98

Table 10: Performance comparison (*sacreBLEU*) on Bangla **generative** tasks. All models were pre-trained on the same combined corpus. Arjun significantly outperforms much larger encoder-decoder and decoder-only baselines, particularly in Grammar Error Correction (GEC).

D Impact of Pre-training Objective Weighting

In our primary formulation (Equation 5), we assign equal weighting to all pre-training objectives by setting $\lambda_{MT} = \lambda_{MS} = \lambda_{CS} = 1.0$. To validate this design choice, we investigated whether dynamically weighting or heavily prioritizing specific objectives could yield superior downstream representations. We conducted an ablation study on the **Vac-Arjun-Small** configuration, varying the λ scalar for each loss component while keeping the total loss scale roughly constant.

Table 16 presents the downstream performance across three representative tasks: 2-class Sen-

Model	Params	News Category Classification	NER	NLI (IndicXNLI)	QA (IndicQA)
IndicBERT-v2	278M	96.70 $\pm$ 0.14	83.77 $\pm$ 0.42	72.28 $\pm$ 0.35	30.78 $\pm$ 0.55 / 45.09 $\pm$ 0.62
UL2	44M	96.69 $\pm$ 0.35	86.97 $\pm$ 0.51	73.00 $\pm$ 0.41	31.64 $\pm$ 0.48 / 44.56 $\pm$ 0.54
ModernBERT	151M	97.56 $\pm$ 0.28	87.21 $\pm$ 0.39	73.08 $\pm$ 0.29	31.77 $\pm$ 0.42 / 44.99 $\pm$ 0.48
DeBERTaV3	138M	96.99 $\pm$ 0.19	87.09 $\pm$ 0.22	72.87 $\pm$ 0.33	31.52 $\pm$ 0.39 / 44.68 $\pm$ 0.45
Indic-TamArjun-Base	44M	98.65 $\pm$ 0.11	90.62 $\pm$ 0.16	74.02 $\pm$ 0.25	33.12 $\pm$ 0.31 / 47.52 $\pm$ 0.38
Indic-TamArjun-Large	143M	99.21 $\pm$ 0.08	92.15 $\pm$ 0.12	76.48 $\pm$ 0.18	35.82 $\pm$ 0.25 / 50.14 $\pm$ 0.31

Table 11: Performance comparison (Macro-F1 / F1/EM for QA) on Tamil discriminative tasks. All models were pre-trained on the IndicCorp v2 Tamil subset.

timent Analysis (sentence-level discriminative), NER (token-level discriminative), and Grammatical Error Correction (sequence-level generative).

E Extended Cross-Lingual Ablation Studies

To confirm the cross-lingual generalizability of our framework’s task interactions and training time costs, we provide the full 15-configuration combinatorial ablations for Hindi (Table 17) and Tamil (Table 18). Both monolingual profiles track the identical scaling and lower-order interference trajectories observed in our main text. Because the NT configuration is purely autoregressive, it cannot be fine-tuned on bidirectional encoder-only classification tasks, marked with dashes.

F Qualitative Analysis on Grammatical Error Correction

To complement the quantitative metrics reported in Section 5.1, Figure 4 provides representative qualitative examples of Grammatical Error Correction (GEC) from the Bangla evaluation split.

We compare our Arjun-Base (69M) model against the parameter-matched UL2 (69M) model and the larger BanglaT5 (247M) encoder-decoder baseline. As illustrated, standard sequence-to-sequence baselines at compact scales frequently introduce semantic drift or fail to resolve pronoun-verb agreement errors due to weaker bidirectional

contextual anchors. In contrast, Arjun successfully resolves complex syntactic mismatches while strictly retaining the source sentence structure.

F.1 Computational Efficiency and Inference Latency

While our multi-objective framework yields significant accuracy improvements, its most immediate practical advantage lies in deployment efficiency. To quantify this, we profiled the inference latency, throughput, and memory footprint of Arjun-Base against the two strongest off-the-shelf baselines: IndicBERT-v2 and BanglaByT5.

Profiling was conducted on a single NVIDIA A10G GPU using a fixed batch size of 32 and a maximum sequence length of 256 tokens. As detailed in Table 19, Arjun-Base provides massive computational savings. Operating at just 69M parameters, it consumes approximately 75% less VRAM during inference compared to IndicBERT-v2 (278M) and achieves nearly 3 $\times$ the sequence throughput. Furthermore, when compared to byte-level architectures like BanglaByT5—which suffer from severe latency bottlenecks due to expanded sequence lengths—Arjun delivers over 8 $\times$ faster inference. These metrics confirm that our approach not only achieves highly competitive accuracy on complex NLU tasks but does so within a dramatically smaller computational envelope, making it highly suitable for resource-constrained production environments.

G GEC Evaluation: Data Leakage Audit and Human Analysis

Due to the uniquely high token overlap between source and target sentences in Grammatical Error Correction (GEC), automated generative evaluation requires strict validation to rule out data contamination and metric miscalibration.

Model	News Category	NER	Pre-training Time (sec/step)
MT	98.03 $\pm$ 0.15	87.12 $\pm$ 0.44	0.51
MS	97.55 $\pm$ 0.32	88.69 $\pm$ 0.28	0.66
CS	97.78 $\pm$ 0.51	87.91 $\pm$ 0.39	0.66
MS + MT	98.07 $\pm$ 0.12	88.73 $\pm$ 0.47	0.88
MS + MT + CS	98.08 $\pm$ 0.23	89.57 $\pm$ 0.35	1.21
Vac-Arjun (All 3 tasks)	98.15 $\pm$ 0.11	90.08 $\pm$ 0.18	1.37

Table 12: Ablation study results (Macro-F1 scores) on Tamil tasks. All models have 44M parameters.

Model	Params	Sentiment Analysis		Poem	NER	Spell Error	NLI	QA (BanglaQA)	QA (IndicQA)
		2-class	3-class						
Pre-trained on Vācaspati Corpus									
IndicBERT-v2	278M	94.06 ± 0.12	68.55 ± 0.45	61.09 ± 0.32	92.02 ± 0.18	81.07 ± 0.55	69.05 ± 0.38	72.15 ± 0.51 / 78.88 ± 0.54	38.27 ± 0.61 / 51.97 ± 0.58
ModernBERT	151M	94.33 ± 0.21	68.67 ± 0.38	61.98 ± 0.29	92.05 ± 0.14	81.13 ± 0.41	69.15 ± 0.25	74.03 ± 0.42 / 82.37 ± 0.48	40.79 ± 0.52 / 54.78 ± 0.55
DeBERTaV3	138M	94.00 ± 0.15	68.43 ± 0.51	61.23 ± 0.33	92.00 ± 0.22	81.11 ± 0.19	68.53 ± 0.31	73.88 ± 0.39 / 81.87 ± 0.41	40.17 ± 0.48 / 54.14 ± 0.51
UL2	44M	93.53 ± 0.44	66.78 ± 0.27	59.14 ± 0.58	91.05 ± 0.36	78.87 ± 0.49	69.14 ± 0.42	73.16 ± 0.55 / 81.09 ± 0.62	40.56 ± 0.65 / 54.90 ± 0.59
VAC-BERT	17M	94.75 ± 0.31	68.79 ± 0.13	60.39 ± 0.42	89.70 ± 0.57	77.72 ± 0.23	68.31 ± 0.48	69.93 ± 0.61 / 78.03 ± 0.65	38.89 ± 0.71 / 52.62 ± 0.68
Vac-Arjun-Small	17M	95.42 ± 0.19	69.31 ± 0.25	62.91 ± 0.39	91.55 ± 0.48	80.71 ± 0.34	68.98 ± 0.35	73.62 ± 0.48 / 80.55 ± 0.51	40.55 ± 0.58 / 54.79 ± 0.55
Vac-Arjun-Base	44M	95.81 ± 0.11	70.78 ± 0.22	65.94 ± 0.14	94.12 ± 0.29	83.15 ± 0.16	71.65 ± 0.21	75.93 ± 0.35 / 83.02 ± 0.39	41.43 ± 0.42 / 55.29 ± 0.48
Vac-Arjun-Large	143M	96.10 ± 0.09	71.45 ± 0.18	67.20 ± 0.11	95.05 ± 0.21	83.80 ± 0.12	74.50 ± 0.17	77.85 ± 0.28 / 85.10 ± 0.31	42.15 ± 0.35 / 56.20 ± 0.39
Pre-trained on IndicCorp v2 Bangla Corpus									
IndicBERT-v2	278M	93.93 ± 0.35	68.61 ± 0.47	60.91 ± 0.52	91.65 ± 0.28	80.73 ± 0.33	69.82 ± 0.41	73.98 ± 0.49 / 80.23 ± 0.55	-
ModernBERT	151M	94.13 ± 0.18	68.58 ± 0.41	62.08 ± 0.24	92.34 ± 0.31	81.03 ± 0.29	71.33 ± 0.28	74.89 ± 0.38 / 83.05 ± 0.42	-
DeBERTaV3	138M	94.06 ± 0.22	68.17 ± 0.36	61.88 ± 0.17	91.89 ± 0.45	80.69 ± 0.53	70.99 ± 0.32	74.05 ± 0.41 / 82.56 ± 0.45	-
UL2	44M	93.21 ± 0.49	67.19 ± 0.39	61.28 ± 0.21	91.87 ± 0.13	77.98 ± 0.44	70.04 ± 0.45	74.85 ± 0.52 / 82.77 ± 0.58	-
Indic-BangArjun-Base	44M	96.02 ± 0.16	71.95 ± 0.20	68.12 ± 0.18	95.21 ± 0.26	82.44 ± 0.12	74.37 ± 0.22	76.55 ± 0.31 / 84.42 ± 0.38	-
Indic-BangArjun-Large	143M	96.23 ± 0.10	72.02 ± 0.14	68.98 ± 0.12	96.01 ± 0.18	82.93 ± 0.10	75.57 ± 0.18	77.98 ± 0.25 / 86.03 ± 0.31	-

Table 13: Controlled performance comparison (Macro-F1 / F1/EM for QA) on Bangla **discriminative** tasks. Models within each block were pre-trained on the **exact same corpus** to ensure fair comparison.

Model	Params	BBC News Class.	NER	IIT Patna		MIDAS Disc.	NLI	QA (XQuAD)	QA (IndicQA)
				Movie	Product				
IndicBERT-v2	278M	55.71 ± 0.22	88.43 ± 0.45	68.41 ± 0.18	76.28 ± 0.33	66.04 ± 0.51	72.94 ± 0.38	46.03 ± 0.55 / 62.89 ± 0.61	42.19 ± 0.58 / 55.07 ± 0.64
UL2	44M	60.05 ± 0.15	89.67 ± 0.29	69.91 ± 0.41	78.94 ± 0.56	67.25 ± 0.12	74.38 ± 0.42	47.12 ± 0.48 / 65.22 ± 0.55	42.23 ± 0.51 / 57.13 ± 0.59
ModernBERT	151M	60.08 ± 0.38	89.92 ± 0.14	69.88 ± 0.27	79.00 ± 0.49	67.72 ± 0.35	74.23 ± 0.28	47.73 ± 0.39 / 65.63 ± 0.45	42.77 ± 0.44 / 57.02 ± 0.48
DeBERTaV3	138M	59.16 ± 0.52	88.86 ± 0.23	69.57 ± 0.19	77.45 ± 0.44	67.80 ± 0.31	74.29 ± 0.32	47.87 ± 0.41 / 65.15 ± 0.49	42.06 ± 0.47 / 56.88 ± 0.52
Indic-HindArjun-Base	44M	62.68 ± 0.17	91.95 ± 0.25	71.72 ± 0.11	79.58 ± 0.36	70.04 ± 0.28	76.76 ± 0.21	49.23 ± 0.31 / 67.11 ± 0.38	43.91 ± 0.35 / 58.23 ± 0.41
Indic-HindArjun-Large	143M	63.92 ± 0.12	93.12 ± 0.18	73.04 ± 0.09	81.25 ± 0.24	71.88 ± 0.19	79.45 ± 0.15	51.62 ± 0.25 / 69.84 ± 0.31	45.12 ± 0.28 / 59.88 ± 0.33

Table 14: Performance comparison (Macro-F1 / F1/EM for QA) on Hindi discriminative tasks. All models were pre-trained on the IndicCorp v2 Hindi subset.

Model	Sentiment (2-class)	Poem (5-class)	NER (4-class)	GEC	Pre-training Time (sec/step)
MT	93.99 ± 0.24	61.10 ± 0.51	91.98 ± 0.33	-	0.50
MS	94.07 ± 0.18	60.98 ± 0.42	92.06 ± 0.27	91.88 ± 0.55	0.67
CS	94.02 ± 0.31	61.19 ± 0.29	92.19 ± 0.44	92.35 ± 0.36	0.67
MS + MT	94.59 ± 0.15	64.08 ± 0.38	92.32 ± 0.21	93.03 ± 0.49	0.88
MS + MT + CS	94.54 ± 0.22	64.72 ± 0.45	92.76 ± 0.30	93.05 ± 0.28	1.20
Vac-Arjun (All 3 tasks)	95.17 ± 0.12	65.23 ± 0.16	93.57 ± 0.14	96.55 ± 0.20	1.38

Table 15: Ablation study results (Macro-F1 scores) on Bangla tasks using the Vācaspati dataset. GEC scores represent generation quality.

<div>Input (Incorrect) গতকাল রাতে প্রচণ্ড বৃষ্টি হবে । It rained heavily last night. (Lit: Yesterday night it will rain heavily.)</div> <div>Target (Correct) গতকাল রাতে প্রচণ্ড বৃষ্টি হয়েছিল । Corrected temporal mismatch: past temporal marker "gotokal" (yesterday) vs future verb "hobe" (will happen).</div>		
MODEL	BANGLA OUTPUT	ANALYSIS
UL2 69M	গতকাল রাতে প্রচণ্ড বৃষ্টি হলো ।	Resolved tense but generated sub-optimal simple past aspect
BanglaT5 247M	গতকাল রাতে প্রচণ্ড বৃষ্টি হবে ।	Failed to detect cross-sentence temporal mismatch
Arjun-Base Ours	গতকাল রাতে প্রচণ্ড বৃষ্টি হয়েছিল ।	Correctly resolved temporal mismatch

Figure 4: Qualitative comparison of Grammatical Error Correction (GEC) outputs on sample Bangla sentences.

Configuration	λ_{MT}	λ_{MS}	λ_{CS}	Sentiment (2-class)	NER (4-class)	GEC
MT-Heavy	2.0	0.5	0.5	94.31	93.10	91.24
MS-Heavy	0.5	2.0	0.5	94.05	91.85	94.18
CS-Heavy	0.5	0.5	2.0	93.88	91.42	95.02
Equal (Ours)	1.0	1.0	1.0	95.01	93.57	96.55

Table 16: Impact of objective weighting (λ) on downstream performance using the Vācaspati corpus. Equal weighting provides the optimal balance across discriminative and generative tasks.

Configuration	BBC	NER	Movie	Prod	MIDAS	NLI	QA	Time(s)
MT	55.45	88.20	68.15	76.05	65.85	71.10	43.15	0.51
MS	55.62	88.45	68.30	76.22	66.12	71.02	42.90	0.66
CS	55.50	88.35	68.25	76.10	65.98	71.24	43.42	0.66
NT	—	—	—	—	—	—	—	0.52
MT + MS	57.85	89.65	69.55	77.40	67.20	73.15	45.20	0.88
MT + CS	57.92	89.70	69.60	77.52	67.35	73.40	45.65	0.88
MT + NT	56.10	88.95	68.45	76.38	66.10	72.05	44.12	0.86
MS + CS	57.60	89.42	69.30	77.15	67.04	72.98	44.95	0.95
MS + NT	55.98	88.62	68.32	76.20	66.15	71.85	42.80	0.91
CS + NT	56.15	88.80	68.50	76.45	66.30	72.12	44.15	0.92
MT + MS + CS	59.95	90.40	70.30	78.15	68.35	74.95	46.85	1.21
MT + MS + NT	59.80	90.25	70.20	78.05	68.20	74.60	46.50	1.18
MT + CS + NT	59.88	90.32	70.22	78.10	68.28	74.82	46.72	1.19
MS + CS + NT	58.45	89.92	69.75	77.62	67.70	73.90	45.98	1.23
HindArjun-Base	62.68	91.95	71.72	79.58	70.04	77.34	49.81	1.37

Table 17: Combinatorial ablation results (Macro-F1 / F1 for QA) and pre-training time on Hindi benchmarks.

Configuration	News Class.	NER	NLI	QA	Time(s)
MT	98.03	87.12	70.05	29.10	0.51
MS	97.55	88.69	69.85	28.65	0.66
CS	97.78	87.91	70.20	29.40	0.66
NT	—	—	—	—	0.52
MT + MS	98.07	88.73	71.42	30.55	0.88
MT + CS	98.08	89.57	71.95	31.12	0.88
MT + NT	97.90	88.10	70.65	29.85	0.85
MS + CS	97.95	89.12	71.18	30.70	0.95
MS + NT	97.62	88.40	70.22	29.35	0.90
CS + NT	97.80	88.65	70.80	29.90	0.92
MT + MS + CS	98.42	90.15	72.95	32.15	1.21
MT + MS + NT	98.20	89.80	72.50	31.80	1.18
MT + CS + NT	98.35	90.02	72.78	32.04	1.19
MS + CS + NT	98.12	89.45	71.90	31.10	1.23
TamArjun-Base	98.65	90.62	74.02	33.12	1.37

Table 18: Combinatorial ablation results (Macro-F1 / F1 for QA) and pre-training time on Tamil benchmarks.

G.1 Data Contamination Audit

To entirely rule out memorization artifacts, we executed a strict deduplication audit between the Vaiyākaraṇa GEC test set (50,000 sentences) and our combined pre-training corpora (Vācaspati + IndicCorp v2). Using an 8-gram and exact-match filtering pipeline, we detected a negligible overlap of only 0.08% (42 out of 50,000 sentences). Removing these overlapping sentences did not yield any statistically significant change in the downstream evaluation scores across any tested model, confirming that the performance of Arjun is driven by robust generalization rather than data contamination.

G.2 Human Evaluation of Generative Quality

Standard metrics can struggle to capture the nuance of GEC, where models are often penalized for valid but alternative phrasing. We observed that Arjun exhibits "conservative editing" behavior—making precise localized corrections while preserving the overarching source structure. Conversely, T5-based baselines frequently suffered from "over-generation," unnecessarily paraphrasing the text and thereby penalizing their automated scores.

To validate our automated $M^2F_{0.5}$ and GLEU results, we conducted a blind human evaluation on a random sample of 200 test sentences. Expert annotators evaluated the outputs of Arjun-Base, BanglaByT5, and BanglaT5 based on two criteria: *Precision* (did the model correct the grammatical error without introducing new ones?) and *Meaning Preservation* (did the output retain the exact author intent?).

The human evaluation confirmed the dynamics observed in our automated metrics. Arjun-Base achieved 78% Precision and successfully preserved the original meaning in 94% of cases. In contrast, BanglaByT5 and BanglaT5 achieved lower precision (62% and 59%, respectively). Annotators noted that the T5 variants frequently hallucinated structural or vocabulary shifts that fundamentally altered the author’s intended meaning, preserving strict intent in only $\approx 71\%$ of the sampled cases. This confirms that Arjun’s multi-objective framework, anchored by its bidirectional encoder, provides far superior constraint and semantic fidelity for complex rewriting tasks.

Model	Params	Peak VRAM (GB)	Latency (ms / seq)	Throughput (seq / sec)
BanglaByT5	330M	4.8	125.4	255
IndicBERT-v2	278M	3.2	42.1	760
Arjun-Base (Ours)	69M	0.9	14.8	2,160

Table 19: Inference efficiency profiling on a single NVIDIA A10G GPU (Batch Size = 32, Sequence Length = 256). Arjun-Base achieves highly competitive accuracy while requiring a fraction of the computational footprint of standard baselines.