BLP 2025

The Second Workshop on Bangla Language Processing (BLP-2025)

Proceedings of the Workshop

December 23, 2025

The BLP organizers gratefully acknowledge the support from the following sponsors.

Sponsor

BLACKBIRD.AI

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-314-2

Preface

Welcome to the Second Workshop on Bangla Language Processing (BLP 2025), collocated with IJCNLP-AACL 2025 and hosted in Mumbai, India.

In this edition, the program is rich and varied, featuring a keynote talk, four paper presentation sessions, and one poster session.

For the regular workshop track, we received 55 submissions. Each paper was rigorously peer-reviewed by two to three expert reviewers in the field. From these submissions, 32 papers were accepted, 20 of which were selected for oral presentation. We note that this distinction reflects scheduling and thematic considerations rather than any difference in quality between oral and poster presentations.

The workshop also featured two shared tasks: (1) Task 1: Hate Speech Detection, and (2) Task 2: Code Generation. Both tasks were well received, with robust participation. For Task 1, we had 161, 103, and 90 team registrations for Subtasks 1A, 1B, and 1C, respectively, culminating in 20 system description papers. For Task 2, we had registrations from 71 teams, culminating in 16 system description papers.

Each system description paper for the shared tasks was peer-reviewed by at least two expert reviewers. The proceedings include these system papers along with two comprehensive overview papers, which will be presented in an oral session at the workshop.

We were fortunate to secure sponsorship funding for the workshop, which has been instrumental in subsidizing registrations for students and aspiring young researchers.

Finally, we would like to thank all contributing authors and the 91 members of the Program Committee for their dedication and for providing high-quality reviews in a timely manner.

Firoj Alam, Md Tahmid Rahman Laskar, Sudipta Kar, and Shammur Absar Chowdhury
On behalf of the BLP Workshop Organizing Committee
Workshop website: `https://blp-workshop.github.io/`

Organizing Committee

Organizers

Firoj Alam, Qatar Computing Research Institute, Qatar
Sudipta Kar, Oracle, USA
Shammur Absar Chowdhury, Qatar Computing Research Institute, Qatar
Naeemul Hassan, University of Maryland, USA
Enamul Hoque, York University, Canada
Md Tahmid Rahman Laskar, York University and Dialpad Inc., Canada
Tasnim Mohiuddin, Qatar Computing Research Institute, Qatar
Md Rashad Al Hasan Rony, Deutsche Telekom, Germany

Shared Task Organizers (Task 1)

Md Arid Hasan, University of Toronto, Canada
Firoj Alam, Qatar Computing Research Institute, Qatar
Md Fahad Hossain, Daffodil International University, Bangladesh
Usman Naseem, Macquarie University, Australia
Syed Ishtiaque Ahmed, University of Toronto, Canada

Shared Task Organizers (Task 2)

Nishat Raihan, George Mason University, USA
Pranav Gupta, Lowe's, USA
Mezbaur Rahman, University of Illinois Chicago, USA
Anas Jawad, University of Illinois Chicago, USA
Noshin Ulfat, IQVIA, USA

Program Committee

Reviewers

Abul Hasnat, Ecole Centrale de Lyon (France)
Adnan Ahmad, Technische Universität Berlin (Germany)
Amran Bhuiyan, York University (Canada)
Avijit Mitra, University of Massachusetts, Amherst (United States)
Avisha Das, University of Houston (United States)
Bharti Goel, F5 networks
Biddut Sarker Bijoy, State University of New York at Stony Brook (United States)
Bo Chen, University of Hong Kong
Debanjan Ghosh, Analog Devices (United States)
Dhiman Goswami, George Mason University (United States)
Ercong Nie, Ludwig-Maximilians-Universität München (Germany)
Fardin Ahsan Sakib, George Mason University (United States)
Farhan Tanvir, Georgia State University (United States)
Farzana Islam Adiba, University of Delaware (United States)
G M Shahariar, University of California, Riverside (United States of America)
Geetanjali Rakshit, UC Santa Cruz (United States)
Israt Jahan, York University (Canada)
Kallol Naha, University of Idaho (United States)
Khondoker Ittehadul Islam, University of Groningen (Netherlands)
Krishno Dey, University of New Brunswick (Canada)
Labiba Jahan, Southern Methodist University (United States)
Madhusudan Basak, Dartmouth College (United States)
Mahak Shah, Splunk (United States)
Md Main Uddin Rony, Bowling Green State University (United States)
Md Mushfiqur Rahman, George Mason University (United States)
Md Rashad Al Hasan Rony, BMW Group (Germany)
Md Rizwan Parvez, Qatar Computing Research Institute (Qatar)
Md Saiful Islam, Shahjalal University of Science and Technology (Bangladesh)
Md Sultanul Islam Ovi, George Mason University (USA)
Md Talha Mohsin, University of Tulsa
Md Tanvirul Alam, Rochester Institute of Technology (United States)
Md Towhidul Absar Chowdhury, Microsoft (United States)
Md. Asif Haider, University of California Irvine (United States of America)
Md. Faiyaz Abdullah Sayeedi, United International University (UIU) (Bangladesh)
Md. Mahadi Hassan, University of Central Florida (United States)
Md. Rafiul Biswas, Hamad Bin Khalifa University (HBKU) (Qatar)
Md. Sanzidul Islam, King Abdul Aziz University (Saudi Arabia)
Mehedi Hasan Bijoy, Aalto University (Finland)
Mohammad Azam Khan, Korea Advanced Institute of Science and Technology (South Korea)
Mohammed Saidul Islam, York University (Canada)
Mohammod Akib Khan, BRAC University (Bangladesh)
Muhammad Rafsan Kabir, North South University (Bangladesh)
Nasheen Nur, Florida Institute of Technology (United States)
Navid Ayoobi, University of Houston (United States)
Nikhil Madaan, LinkedIn
Omar Sharif, Dartmouth College (United States)

Rabindra Nath Nandi, Verbex.ai (Bangladesh)
Raian Rahman, Islamic University of Technology (Bangladesh)
Ridwan Mahbub, York University (Canada)
Sabbir Ahmed, Islamic University of Technology (Bangladesh)
Sabit Hassan, University of Pittsburgh (United States)
Sadat Shahriar, University of Houston (United States)
Sadid A. Hasan, Microsoft (United States)
Sadiya Sayara Chowdhury Puspo, George Mason University (United States)
Shamik Roy, Amazon (United States)
Sheng Gao, Quant (United States)
Shervin Malmasi, Amazon (United States)
Shubhashis Roy Dipta, University of Maryland (United States)
Sivaji Bandyopadhyay, Jadavpur University (India)
Somnath , University of Tartu (Estonia)
Sourav Saha, University of Central Florida (United States)
Souvika Sarkar, Wichita State University (United States)
Subrata Ashe, Meta (United States)
Surendrabikram Thapa, Virginia Polytechnic Institute and State University (United States)
Syeda Jannatus Saba, State University of New York at Stony Brook (United States)
Tashin Ahmed, Cross Compass (Japan)
Tirtho Roy, Iowa Sate University (United States)
Wasi Uddin Ahmad, NVIDIA (United States)
Yassine El Kheir, German Research Center for AI (Germany)
Yu Wu, Rutgers University (United States)
Zabir Al Nazi, University of California Riverside (United States)
Zeerak Talat, University of Edinburgh (United Kingdom)

Keynote Talk

Data’r Panchali: The Changing Landscape of Bangla NLP

Monojit Choudhury
2025-12-23 09:10:00 - 09:50:00 –

Abstract: Over the past three decades, Natural Language Processing technologies in Bangla have made remarkable progress. The advent of generative AI has accelerated this journey, leading to impressive breakthroughs across multiple domains. Yet, two profound questions persist. First, what tangible benefits do these advances bring to Bangla speakers? Who stands to gain and who remains excluded? Second, has the technological gap between English (and other high-resource languages) and Bangla truly narrowed, or has it quietly widened?

Ultimately, both the triumphs and the shortcomings of Bangla NLP trace back to one fundamental factor: the availability and absence of data. In this talk, I will explore these questions through both empirical analysis and personal reflection, examining how data shapes access, equity, and innovation. I will conclude with a discussion of some intriguing possibilities: how generative AI might illuminate deeper questions about culture, linguistic diversity, and the evolving identity of Bangla itself.

Bio: Monojit Choudhury is a Professor of Natural Language Processing at the Mohamed bin Zayed University of Artificial Intelligence (MBZUAI) in Abu Dhabi. His research sits at the intersection of language technology and society, with a particular focus on how foundation models learn and (mis)represent linguistic and cultural diversity, and how to design fair, inclusive language technologies for low-resource and marginalized languages. Prior to joining MBZUAI, he was a principal researcher at Microsoft Research India from 2009 to 2022 and a principal applied scientist at Microsoft Turing (part of Microsoft India Development Center) from 2022 to 2023. He also serves as adjunct faculty at the International Institute of Information Technology, Hyderabad (since 2017). Professor Choudhury is the general chair of the Panini Linguistics Olympiad (India’s national linguistics Olympiad) and founding co-chair of the Asia Pacific Linguistics Olympiad. He is deeply committed to popularizing linguistics and natural language processing among schoolchildren and non-experts, often through carefully designed puzzles and problem-solving activities.

Table of Contents

Program

Tuesday, December 23, 2025

09:00 - 09:10 *Opening Remarks*

09:10 - 09:50 *Invited Talk: Data'r Panchali: The Changing Landscape of Bangla NLP*

09:50 - 10:26 *Oral Presentation I (long papers)*

Overview of BLP-2025 Task 1: Bangla Hate Speech Identification
Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem and Syed Ishtiaque Ahmed

Overview of BLP-2025 Task 2: Code Generation in Bangla
Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Santu Karmaker and Marcos Zampieri

BanHateME : Understanding Hate in Bangla Memes thorough Detection, Categorization, and Target Profiling
Md Ayon Mia and Md Fahim

10:30 - 11:00 *Coffee Break*

11:00 - 12:00 *Poster Session*

12:00 - 13:00 *Lunch Break*

13:00 - 14:00 *Oral Presentation II (long papers)*

LP-FT-LoRA: A Three-Stage PEFT Framework for Efficient Domain Adaptation in Bangla NLP Tasks
Tasnimul Hossain Tomal, Anam Borhan Uddin, Intesar Tahmid, Mir Sazzat Hossain, Md Fahim and Md Farhad Alam Bhuiyan

Human–LLM Benchmarks for Bangla Dialect Translation: Sylheti and Chittagonian on the BanglaCHQ-Summ Corpus
Nowshin Mahjabin, Ahmed Shafin Ruhan, Mehreen Chowdhury, Md Fahim and MD Azam Hossain

LLMs for Low-Resource Dialect Translation Using Context-Aware Prompting: A Case Study on Sylheti
Tabia Tanzin Prama

Tuesday, December 23, 2025 (continued)

GRASP-ChoQ: Knowledge Graph-Based Retrieval Augmentation for Stance Detection in Political Texts with Chain-of-Questions Reasoning
Rasel Mahmud, Md. Abdur Rakib Mollah, Aninda Kumar Sharma and Omar Faruq Osama

ChakmaBridge: A Five-Way Parallel Corpus for Navigating the Script Divide in an Endangered Language
Md. Abdur Rahman, Md. Tofael Ahmed Bhuiyan and Abdul Kadar Muhammad Masum

14:00 - 14:10 *Break*

14:10 - 15:34 *Oral Presentation III (long papers)*

Form-aware Poetic Generation for Bangla
Amina, Abdullah, Mueeze Al Mushabbir and Sabbir Ahmed

Byte Pair Encoding Is All You Need For Automatic Bengali Speech Recognition
Ahnaf Mozib Samin

Zero-Shot Multi-Label Classification of Bangla Documents: Large Decoders Vs. Classic Encoders
Souvika Sarkar, Md Najib Hasan and Santu Karmaker

P6Jiggasha: Benchmarking Large Language Models on Bangla Physics Question Answering with Cross-lingual Evaluation
S.m. Shahriar, Md Tahmid Hasan Fuad, Md Fahim and Md. Azad Hossain

Exploring Cross-Lingual Knowledge Transfer via Transliteration-Based MLM Fine-Tuning for Critically Low-resource Chakma Language
Adity Khisa, Nusrat Jahan Lia, Tasnim Mahfuz Nafis, Zarif Masud, Tanzir Pial, Shebuti Rayana and Ahmedul Kabir

A Hybrid Transformer–Sequential Model for Depression Detection in Bangla–English Code-Mixed Text
Md Siddikul Imam Kawser, Jidan Al Abrar, Mehebub Bin Kabir, Md. Rayhan Chowdhury and Md Ataullah Bahari

BOIGENRE: A Large-Scale Bangla Dataset for Genre Classification from Book Summaries
Rafi Hassan Chowdhury and Rahanuma Ryaan Ferdous

15:35 - 16:00 *Coffee Break*

Tuesday, December 23, 2025 (continued)

16:00 - 16:48 *Oral Presentation IV (long papers)*

BLUCK: A Benchmark Dataset for Bengali Linguistic Understanding and Cultural Knowledge
Daeen Kabir, Minhajur Rahman Chowdhury Mahim, Sheikh Shafayat, Adnan Sadik, Arian Ahmed, Eunsu Kim and Alice Oh

Advancing Subjectivity Detection in Bengali News Articles Using Transformer Models with POS-Aware Features
Md Minhazul Kabir, Kawsar Ahmed, Mohammad Ashfak Habib and Mohammed Moshiul Hoque

Robustness of LLMs to Transliteration Perturbations in Bangla
Fabiha Haider, Md Farhan Ishmam, Fariha Tanjim Shifat, Md Tasmim Rahman Adib, Md Fahim and Md Farhad Alam Bhuiyan

Clustering LLM-based Word Embeddings to Determine Topics from Bangla Articles
Rifat Rahman

17:00 - 17:30 *Awards and Ending Remarks*

Byte Pair Encoding Is All You Need For Automatic Bengali Speech Recognition

Ahnaf Mozib Samin[*]
Queen's University, Canada
University of Groningen, The Netherlands
University of Malta, Malta
ahnaf.samin@queensu.ca

Abstract

Byte pair encoding (BPE) emerges as an effective tokenization method for tackling the out-of-vocabulary (OOV) challenge in various natural language and speech processing tasks. Recent research highlights the dependency of BPE subword tokenization's efficacy on the morphological nature of the language, particularly in languages rich in inflectional morphology, where fewer BPE merges suffice for generating highly productive tokens. Motivated by this, our study empirically identifies the optimal number of BPE tokens for Bengali, a language known for its morphological complexity, thus enhancing out-of-distribution automatic speech recognition (ASR) performance. Experimental evaluation reveals that an excessively high number of BPE tokens can lead to overfitting, while approximately 500-1000 tokens result in superior OOV performance. Furthermore, we conduct a comparative analysis of BPE with character-based and unigram-based tokenization methods. By introducing BPE tokenization to Bengali ASR, we achieve a substantial reduction in the word error rate (WER) from 66.44% in our character-based baseline system to 63.80% on the LB-ASRTD eval set and from 46.34% to 42.80% on the SHRUTI eval set, both of which include out-of-distribution data.

1 Introduction

The performance of an automatic speech recognition system is contingent on its core components including acoustic feature extraction, mapping acoustic features to tokens using the Gaussian Mixture Model/Hidden Markov Model (GMM-HMM) or neural networks, and language model-based rescoring of the outputs from connectionist temporal classification (CTC), etc (Graves et al., 2006). As for segmentation, either word-level or subword-level tokens can be modeled to build ASR systems. However, word-level modeling faces a challenge due to the vast number of words in a language, which exceeds the typical vocabulary size of an ASR system. As a result, word-level modeling is susceptible to the out-of-vocabulary (OOV) problem (Livescu et al., 2012). An OOV word refers to a word that was not encountered during model training but appears during the inference phase.

Subword units are known as effective solutions to tackle the OOV issue in natural language processing (NLP) (Sennrich et al., 2016). Examples of subword units include phonemes, characters, unigrams, and byte pair encoding (BPE) tokens, etc (Kudo, 2018; Gage, 1994; Sennrich et al., 2016). Phonemes represent the smallest units of sound, and after training with phonemes, a model can infer new words. Creating a lexicon that maps each word to its corresponding phonemes, however, requires domain expertise as well as a substantial amount of time and effort for manual annotation (Harwath and Glass, 2014). Character-based ASR models are easier to develop since mapping between words and their corresponding characters can be done automatically (Chan et al., 2016). Furthermore, training a model with a limited number of characters in a language is more computationally efficient than training a word-based model. Unigram language modeling is another segmentation technique that removes tokens based on language model perplexity, initially applied in machine translation (Kudo, 2018).

BPE subword tokenization is first utilized in neural machine translation (NMT) and has gained widespread usage due to their ability to handle OOV words effectively (Gage, 1994; Sennrich et al., 2016; Radford et al., 2018). Subsequent studies implement BPE-based subword modeling in the speech processing domain (Synnaeve et al., 2020; Yusuyin et al., 2023). For BPE, the number of merge operations determines the number of generated tokens/subwords. From the work of Gutierrez-

[*] Work performed during the Erasmus Mundus Joint MSc. program at the University of Groningen and the University of Malta.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 1–7
December 23, 2025 ©2025 Association for Computational Linguistics

Vasques et al. (2023) on 47 diverse languages, it has been found that in languages characterized by extensive inflectional morphology, there is a tendency to generate highly productive subwords during the initial merging steps. Conversely, in languages with limited inflectional morphology, idiosyncratic subwords tend to play a more prominent role (Parra, 2024). Therefore, the characteristics of subwords and the required number of merges in BPE tokenization are contingent upon the morphological nature of the respective language. Moreover, an empirical study is conducted by incrementally increasing the BPE merges going from characters to words (Gutierrez-Vasques et al., 2021). The authors reported that around 200 BPE merges result in the most similar distribution across different languages.

Since the optimal number of BPE merges cannot be universally determined for different languages with varied types of morphology (Gutierrez-Vasques et al., 2023), in this study, we empirically determine the number of BPE merges needed for a highly inflectional language—Bengali to achieve superior out-of-distribution ASR performance. Bengali is an Indo-Aryan language spoken in Bangladesh and India and poses challenges to the development of robust ASR systems due to its intricate morphological forms and inadequate research (Ali et al., 2008; Samin et al., 2021). Subsequently, we compare the results of BPE-based tokenization with segmentation approaches based on characters and unigrams by performing a cross-dataset evaluation, aiming to understand their effectiveness for handling out-of-distribution data. To the best of our knowledge, this investigation exploiting different subword modeling approaches is conducted for the first time for Bengali ASR.

The rest of the paper is structured as follows: a comprehensive background study on BPE and unigram language modeling, along with a review of related work in the speech processing domain is provided in Section 2. The methodology of our experiments are described in Section 3. Details about the experiment setup are provided in Section 4. Results are discussed in Section 5. The conclusion and outlines the future directions are provided in Section 6.

2 Background

2.1 Byte pair encoding

Byte pair encoding is a data compression algorithm, which was applied in NMT in 2016 (Gage, 1994; Sennrich et al., 2016).

Algorithm 1 Byte-pair encoding (Gage, 1994; Sennrich et al., 2016; Bostrom and Durrett, 2020)

```
$S \leftarrow$ set of strings (Approx. 40k Bengali words)
$n \leftarrow$ target vocab size
procedure BPE($S, n$)
    $V \leftarrow$ all unique characters in $S$
    while $|V| < n$ do
        Step 1 Merge tokens $a$ and $b$, where
            $a, b \in V$ and represent the most
            frequent bigram in $S$
        Step 2 Create a new token $ab$ by
            concatenating $a$ and $b$
        Step 3 Add $ab$ to $V$
        Step 4 Replace each bigram occurrence
            of $a, b$ tokens in $S$ with $ab$
    end while
    return $V$
end procedure
```

BPE algorithm takes a set of strings S and aims to create a vocabulary V with a target size of n. It iteratively merges the most frequent bigram in S into a new token, updating the vocabulary and replacing occurrences of the merged tokens in the original strings. The algorithm continues until the vocabulary size reaches the desired target size n and returns the final vocabulary V.

2.2 Unigram language modeling

Unigram language modeling (LM) was first applied in NMT in 2018 and compared the performance to that of BPE (Kudo, 2018). Unigram language modeling algorithm takes a set of strings S and aims to create a vocabulary V with a target size of n. It starts by initializing V with all substrings occurring more than once in S (without crossing words). The algorithm then iteratively prunes the vocabulary by estimating the token 'loss' L_t for each token t in V using the unigram language model θ. The tokens with the highest L_t values are removed from V until its size reaches the target vocabulary size n. Finally, the algorithm fits the final unigram language model θ to S and returns V and θ as the resulting vocabulary and language model, respectively.

Algorithm 2 Unigram LM (Kudo, 2018; Bostrom and Durrett, 2020)

```
$S \leftarrow$ set of strings (Approx. 40k Bengali words)
$n \leftarrow$ target vocab size
procedure UNIGRAM($S, n$)
    $V \leftarrow$ all substrings occurring more than
        once in $S$ (not crossing words)
    while $|V| > n$ do
        Build the unigram language model $\theta$
        with $S$
        for $t$ in $V$ do
            $L_t \leftarrow pplx_\theta(S) - pplx_{\theta'}(S)$
            where $\theta'$ is the LM without token t
        end for
        Remove min($|V| - n$, $\lfloor\alpha|V|\rfloor$) of the
        tokens $t$ with highest $L_t$ from $V$ ,
        where $\alpha \in$ [0, 1] is a hyperparameter
    end while
    Build final unigram LM $\alpha$ to $S$
    return $V, \theta$
end procedure
```

Though both BPE and unigram language modeling are subword tokenization algorithms that produce a fixed-size vocabulary for text segmentation, there is a key difference in the method. BPE iteratively merges the most frequent adjacent symbol pairs in the corpus to form new tokens, following a deterministic and greedy approach. In contrast, unigram language modeling initializes with a large vocabulary of candidate substrings and prunes tokens based on their contribution to the overall likelihood under a probabilistic unigram language model, removing those that decrease the model's likelihood until the target vocabulary size is reached.

2.3 Related work

The choice of subword units for acoustic modeling can depend on settings such as high-variability spontaneous speech, noisy environment, low-resource scenario, or cross-lingual speech recognition (Livescu et al., 2012). Different subword modeling techniques have been explored in numerous studies including improved word boundary marker in weighted finite state transducer (WFST)-based decoder for Finnish and Estonian (Smit et al., 2017), pronunciation-assisted subword modeling (PASM) (Xu et al., 2019), acoustic data-driven subword modeling (ADSM) (Zhou et al., 2021), among others. While both PASM and ADSM are reported to outperform BPE-based modeling for ASR, these two approaches are evaluated with only a morphologically poor language English. Thus, it is uncertain how different subword modeling approaches will work for morphologically rich languages.

More recently, phone-based BPE has been introduced for multilingual speech recognition (Yusuyin et al., 2023). However, in a monolingual setting, PBPE obtains similar performance compared to BPE while both BPE and PBPE outperform phone and character-based modeling.

In recent years, Bengali ASR research has primarily focused on addressing the scarcity of datasets through resource development initiatives (Kjartansson et al., 2018; Ahmed et al., 2020; Kibria et al., 2022; Rakib et al., 2023; Samin et al., 2024). Sadeq et al. (2020) addressed the challenge of manual annotation in training ASR systems by proposing a semi-supervised approach for Bangla ASR, leveraging large unpaired audio and text data encoded in an intermediate domain with a novel loss function. Samin et al. (2021) evaluated the LB-ASRTD corpus (Kjartansson et al., 2018), a large-scale publicly available dataset comprising 229 hours, utilizing deep learning-based methods and performing a character-wise error analysis. While earlier studies on Bengali ASR involved phone-based segmentation (Al Amin et al., 2019), the shift towards end-to-end ASR systems has made character-based models more prevalent (Samin et al., 2021). Notably, to the best of our knowledge, there has been no study comparing different subword modeling techniques in the Bengali speech processing domain.

3 Method

We train a convolutional neural network (CNN) based acoustic model for performing the experiments. We extract 21 mel-frequency cepstral coefficients (MFCCs) from each frame of the input signal and feed it to the CNN. The frame length and stride are 30 ms and 15 ms, respectively. We implement the same CNN architecture following the work of Samin et al. (2021), except for introducing a batch normalization

layer in each convolution block to improve optimization stability and reduce internal covariate shift. Moreover, we increase the number of convolutional blocks from 15 to 20, enabling the network to learn deeper hierarchical acoustic representations. The objective of the acoustic model is to predict the subword units based on the CTC loss cri-

Table 1: Six acoustic models are trained with three types of subword units such as character, unigram, and BPE. For BPE segmentation, 500, 1K, 2K, and 3K target tokens are fixed in separate experiments. The models are trained with a CNN architecture on the Bengali SUBAK.KO train set and evaluated on the eval sets of SUBAK.KO, LB-ASRTD, and SHRUTI. TERs (%) and WERs (%) are reported. Bold numbers indicate the best WERs in the corresponding eval sets.

Token type	# tokens	SUBAK.KO eval		LB-ASRTD eval		SHRUTI eval	
		TER	WER	TER	WER	TER	WER
Character	73	5.41	18.89	27.14	66.44	14.31	46.34
Unigram	1000	6.03	16.86	30.32	66.07	15.97	44.40
BPE	500	5.71	17.11	28.15	64.28	14.61	**42.80**
BPE	1000	5.97	16.65	29.32	**63.80**	15.97	43.75
BPE	2000	6.19	16.17	31.99	66.38	17.36	44.58
BPE	3000	6.34	**15.63**	34.11	66.46	18.84	45.77

terion given the audio signal (Graves et al., 2006). We choose either character, BPE, or unigram tokens in individual experiments as our subwords.

We do not perform beam search decoding with a language model since it can have an impact on the final result. The goal of this study is to investigate different subword-based acoustic modeling for a morphologically rich language Bengali, so we exclude the language modeling part. Therefore, greedy search decoding is used to generate the output tokens. In this approach, at each decoding step, the token with the highest predicted probability is selected without considering alternative sequences. This method simplifies decoding and allows us to evaluate the acoustic model's performance independently of any language model influence.

4 Experiment setup

We implement CNN-based acoustic models using the Flashlight toolkit (Kahn et al., 2022). We train our CNNs using SUBAK.KO, an annotated Bangla speech dataset (Kibria et al., 2022). SUBAK.KO is mostly a read speech corpus with 229 hours of read speech and only 12 hours of broadcast speech. We use the same 200-hour long training set, 20-hour long development (dev) set, and 20-hour long evaluation (eval) set following Kibria et al. (2022). Using standard train, dev, and eval sets enables us to compare our strategy to those of the past. For a comprehensive evaluation, we use a 20-hour subset of the large Bangla automatic speech recognition training data (LB-ASRTD) and the 20-hour long full SHRUTI corpus (Kjartansson et al., 2018; Das et al., 2011). Our SUBAK.KO-based ASR model encounters OOV words from LB-ASRTD and SHRUTI out-of-distribution data. Therefore, cross-evaluation assures a more reliable evaluation of various subword modeling algorithms.

We train baseline ASR systems using character and unigram tokens, subsequently contrasting their performance with BPE-based ASR systems. For character-based modeling, we simply use Python programming language to segment a word into individual characters. To build the BPE and unigram-based lexicons, we use the Sentence-piece library (Kudo and Richardson, 2018). For unigram language modeling, we use a fixed token size of 1000. As for BPE, we develop four acoustic models with 500, 1K, 2K, and 3K tokens and compare the results. We use the SUBAK.KO train set as a text corpus to generate the BPE and unigram tokens.

For evaluating ASR models, we employ standard metrics such as the word error rate (WER) and the token error rate (TER). Here, token represents either characters, BPE or Unigram units. Eight graphics processing units (GPUs) with only 12 gigabytes of virtual random access memory each are used to train the models.

5 Results & Discussion

Table 1 presents the WERs and TERs for various subword modeling types and token sizes. On all three evaluation sets, BPE-based acoustic modeling outperforms both character-based and unigram-based modeling in terms of WERs. Unigram modeling achieves lower WERs than character-based segmentation. With the same number of tokens (1000 tokens), unigram modeling cannot surpass the BPE-based approach when dealing with both in-distribution (SUBAK.KO eval) and out-of-distribution (LB-ASRTD and SHRUTI eval) data.

The number of generated tokens is proportional to the number of BPE merge operations. As we increase the number of BPE tokens, the WERs continue to decrease on the SUBAK.KO eval set. Nevertheless, acoustic models trained with 1000 BPE tokens and 500 BPE tokens achieve reduced WERs on the LB-ASRTD and SHRUTI eval sets, respectively. Notably, BPE tokens are generated utilizing the SUBAK.KO train corpus. This implies that as the number of BPE tokens increases, the model becomes overfit on its eval set while performing poorly on the out-of-distribution data. Thus, a target BPE token size of 500 or 1000 is found to be suitable to achieve better generalizability for ASR. This finding conforms to the work of (Gutierrez-Vasques et al., 2023), indicating that morphologically rich languages necessitate fewer BPE merges, leading to a reduced count of BPE tokens. Also, a higher number of BPE merge operations tends to generate longer-length tokens, which resemble words and present similar bottlenecks of word-based tokenization.

With regard to TERs, character-based acoustic modeling exhibits lower error rates than the other two approaches. Furthermore, with the BPE tokens, the TERs tend to continually increase when we increment the number of BPE tokens. Each character represents a token of length one, while BPE tokens undergo multiple merge operations, resulting in longer token lengths. We argue that predicting a token with a shorter length is a comparatively easier task for the acoustic model in contrast to mapping input signal frames to a token with a longer length. This discrepancy can contribute to higher TERs for unigram and BPE-based modeling. However, if a model can accurately predict the long-length tokens of a word, it increases the probability of achieving a lower WER because the number of unigram/BPE tokens in a word is typically fewer than the number of characters in that word.

Figure 1 demonstrates the performance of BPE-based subword modeling with various train corpus sizes. We can observe the positive impact of increasing the amount of acoustic model training data on all three evaluation sets for the BPE-based approach. However, it is worth noting that BPE-based speech recognition systems can still achieve satisfactory performance even in low-resource scenarios.

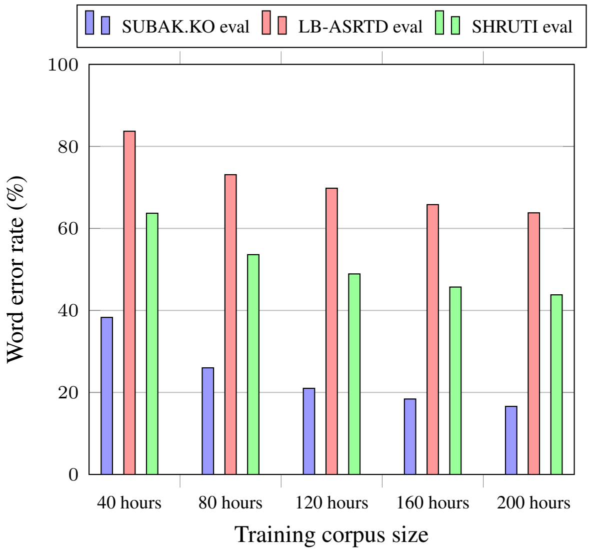


Figure 1: Acoustic models are trained with 1000 BPE tokens on five different SUBAK.KO train subsets (e.g. 40 hours, 80 hours, 120 hours, 160 hours, and 200 hours). SUBAK.KO, LB-ASRTD, and SHRUTI eval sets are used to report the WERs (%)

6 Conclusion & Future Work

In this work, we determine the number of BPE merges in the context of ASR for a morphologically rich language - Bengali and provide intriguing insights into the relationship between BPE merge operations and ASR performance in the presence of OOV words. Furthermore, we provide a comparative analysis for three subword modeling approaches including characters, unigrams, and BPE for ASR. Our empirical study suggests that BPE is a better choice for subword modeling than characters and unigram tokens. Additionally, through cross-dataset evaluation, we find that targeting a token size of approximately 500 or 1000 yields improved generalization and robustness, while excessively high numbers of BPE tokens can result in overfitting. This outcome corresponds with prior linguistic research, suggesting that morphologically rich languages demand fewer BPE merges to yield highly productive BPE tokens, as discussed in (Gutierrez-Vasques et al., 2023).

There are several potential directions for future research. Firstly, instead of generating BPE tokens from the text files of SUBAK.KO train set, a large-scale text corpus could be constructed specifically for this purpose, enabling the generation of BPE tokens from a more extensive and diverse dataset. We hypothesize that this approach could yield superior BPE representations, resulting in enhanced robustness across out-of-domain data, particularly for challenging morphologically rich languages. Secondly, we aim to explore additional morphologically rich languages from diverse language families, as well as languages like English that lack complex morphology, to further evaluate the effectiveness of subword modeling approaches. Lastly, although our study benchmarks convolutional neural networks (CNNs), it would be valuable to investigate state-of-the-art transfer learning algorithms, such as wav2vec 2.0 (Baevski et al., 2020) and HuBERT (Hsu et al., 2021), to determine if BPE subword modeling remains effective with these architectures.

7 Limitations

Although this work investigates BPE subword tokenization method for Bengali ASR for the first time, there are several limitations. First, this study evaluates tokenization only in a monolingual Bengali setting as a representative morphologically rich language. Future work should examine additional such languages to assess generalizability. Second, we investigate the effectiveness of BPE exclusively with CNN-based acoustic models. Exploring alternative architectures remains an open direction. Lastly, this work focuses on comparing BPE with closely related subword approaches (unigram LM and character-based modeling) to isolate and analyze BPE's behavior in morphologically rich Bengali. Therefore, alternative techniques such as PASM (Xu et al., 2019), ADSM (Zhou et al., 2021), and phone-based BPE (Yusuyin et al., 2023), which introduce additional phonetic or acoustic modeling assumptions beyond our scope, are not evaluated in this work.

References

Shafayat Ahmed, Nafis Sadeq, Sudipta Saha Shubha, Md Nahidul Islam, Muhammad Abdullah Adnan, and Mohammad Zuberul Islam. 2020. Preparation of bangla speech corpus from publicly available audio & text. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6586–6592.

Md Alif Al Amin, Md Towhidul Islam, Shafkat Kibria, and Mohammad Shahidur Rahman. 2019. Continuous bengali speech recognition based on deep neural network. In *2019 international conference on electrical, computer and communication engineering (ECCE)*, pages 1–6. IEEE.

Md Nawab Yousuf Ali, SM Abdullah Al-Mamun, Jugal Krishna Das, and Abu Mohammad Nurannabi. 2008. Morphological analysis of bangla words for universal networking language. In *2008 Third International Conference on Digital Information Management*, pages 532–537. IEEE.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964.

Biswajit Das, Sandipan Mandal, and Pabitra Mitra. 2011. Bengali speech corpus for continuous auutomatic speech recognition system. In *2011 International conference on speech database and assessments (Oriental COCOSDA)*, pages 51–55. IEEE.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Ximena Gutierrez-Vasques, Christian Bentz, and Tanja Samardžić. 2023. Languages through the looking glass of bpe compression. *Computational Linguistics*, pages 1–59.

Ximena Gutierrez-Vasques, Christian Bentz, Olga Sozinova, and Tanja Samardzic. 2021. From characters to words: the turning point of bpe merges. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3454–3468.

David F Harwath and James R Glass. 2014. Speech recognition without a lexicon-bridging the gap between graphemic and phonetic systems. In *INTERSPEECH*, pages 2655–2659.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Jacob D Kahn, Vineel Pratap, Tatiana Likhomanenko, Qiantong Xu, Awni Hannun, Jeff Cai, Paden Tomasello, Ann Lee, Edouard Grave, Gilad Avidov, and 1 others. 2022. Flashlight: Enabling innovation in tools for machine learning. In *International Conference on Machine Learning*, pages 10557–10574. PMLR.

Shafkat Kibria, Ahnaf Mozib Samin, M Humayon Kobir, M Shahidur Rahman, M Reza Selim, and M Zafar Iqbal. 2022. Bangladeshi bangla speech corpus for automatic speech recognition research. *Speech Communication*, 136:84–97.

Oddur Kjartansson, Supheakmungkol Sarin, Knot Pipatsrisawat, Martin Jansche, and Linne Ha. 2018. Crowd-sourced speech corpora for javanese, sundanese, sinhala, nepali, and bangladeshi bengali. In *6th Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU 2018)*. ISCA.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Karen Livescu, Eric Fosler-Lussier, and Florian Metze. 2012. Subword modeling for automatic speech recognition: Past, present, and emerging approaches. *IEEE Signal Processing Magazine*, 29(6):44–57.

Iñigo Parra. 2024. Morphological typology in bpe subword productivity and language modeling. In *Latinx in AI@ NeurIPS*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI. OpenAI Technical Report.

Fazle Rabbi Rakib, Souhardya Saha Dip, Samiul Alam, Nazia Tasnim, Md Istiak Hossain Shihab, Md Nazmuddoha Ansary, Syed Mobassir Hossen, Marsia Haque Meghla, Mamunur Mamun, Farig Sadeque, and 1 others. 2023. Ood-speech: A large bengali speech recognition dataset for out-of-distribution benchmarking. *Proc. Interspeech 2023*.

Nafis Sadeq, Nafis Tahmid Chowdhury, Farhan Tanvir Utshaw, Shafayat Ahmed, and Muhammad Abdullah Adnan. 2020. Improving end-to-end Bangla speech recognition with semi-supervised training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1875–1883, Online. Association for Computational Linguistics.

Ahnaf Mozib Samin, M Humayon Kobir, Shafkat Kibria, and M Shahidur Rahman. 2021. Deep learning based large vocabulary continuous speech recognition of an under-resourced language bangladeshi bangla. *Acoustical Science and Technology*, 42(5):252–260.

Ahnaf Mozib Samin, M Humayon Kobir, Md Mushtaq Shahriyar Rafee, M Firoz Ahmed, Mehedi Hasan, Partha Ghosh, Shafkat Kibria, and M Shahidur Rahman. 2024. Banspeech: A multi-domain bangla speech recognition benchmark toward robust performance in challenging conditions. *IEEE Access*, 12:34527–34538.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725. Association for Computational Linguistics (ACL).

Peter Smit, Sami Virpioja, and Mikko Kurimo. 2017. Improved subword modeling for wfst-based speech recognition. In *INTERSPEECH*, pages 2551–2555. International Speech Communication Association.

Gabriel Synnaeve, Qiantong Xu, Jacob Kahn, Tatiana Likhomanenko, Edouard Grave, Vineel Pratap, Anuroop Sriram, Vitaliy Liptchinsky, and Ronan Collobert. 2020. End-to-end asr: from supervised to semi-supervised learning with modern architectures. In *ICML 2020 Workshop on Self-supervision in Audio and Speech*.

Hainan Xu, Shuoyang Ding, and Shinji Watanabe. 2019. Improving end-to-end speech recognition with pronunciation-assisted sub-word modeling. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7110–7114. IEEE.

Saierdaer Yusuyin, Hao Huang, Junhua Liu, and Cong Liu. 2023. Investigation into phone-based subword units for multilingual end-to-end speech recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Wei Zhou, Mohammad Zeineldeen, Zuoyun Zheng, Ralf Schlüter, and Hermann Ney. 2021. Acoustic data-driven subword modeling for end-to-end speech recognition. *arXiv preprint arXiv:2104.09106*.

GRASP-ChoQ: Knowledge Graph-Based Retrieval Augmentation for Stance Detection in Political Texts with Chain-of-Questions Reasoning

Md Rasel Mahmud
USTC, China
raselmahmud@mail.ustc.edu.cn

Md. Abdur Rakib Mollah
Independent Researcher
rakib1703115@gmail.com

Aninda Kumar Sharma
University of Adelaide
aiubanik18@gmail.com

Omar Faruq Osama
SUNY Binghamton
oosama@binghamton.edu

Abstract

Political stance detection in understudied socio-political contexts presents a persistent challenge for language models. It is because dynamic contexts and indirect relationships between political entities complicate the accurate alignment of opinions. To address this, we introduce **GRASP-ChoQ**, an approach that combines structured knowledge graphs with a chain-of-questions reasoning to break down interactions in political texts. We support this with **BPDisC**, a novel dataset of politically charged tweets from Bangladesh during and after the July 2024 protests. Instead of making direct predictions, our approach relies on intermediate reasoning steps facilitated by contextually retrieved subgraphs from the knowledge graph. We evaluated our method on BPDisC and six additional benchmark datasets. Across all datasets, it consistently outperformed baseline LLMs. Notably, DeepSeek R1 combined with GRASP-ChoQ achieved the highest performance on the BPDisC dataset, with a **40%** improvement in F1 score over the zero-shot baseline. These results highlight our method's ability to integrate context and effectively handle complex, low-resource, and evolving political scenarios.[1].

1 Introduction

Since large language models (LLMs) are usually trained on static datasets containing predetermined temporal cutoffs (Liu et al., 2024), they are not aware of events or novel ideas that emerge following the training period. As an example, a pre-trained model might have difficulty identifying a change in political ideology (Liu et al., 2022). Retraining these models by incorporating new data is computationally challenging and often results in updated models losing their previous knowledge. Since LLMs cannot naturally adapt to situations that change quickly, it is hard to figure out political positions in environments that are constantly changing politically and socially (Rozado, 2024).

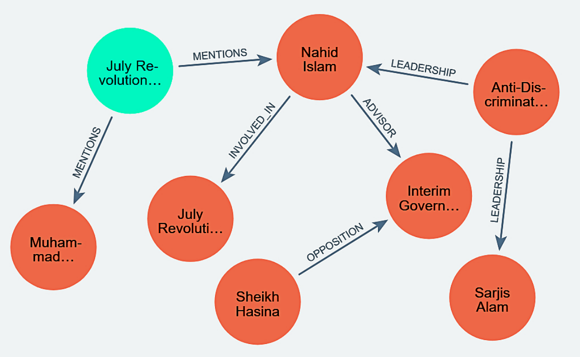


Figure 1: A knowledge graph containing political entities as nodes and their relationships as edges

Identifying political stances often requires understanding indirect relationships—such as informal alliances or ideological disagreements—that are implied rather than explicitly stated in texts (Leifeld and Brandenberger, 2019; Oswald et al., 2021). LLMs struggle to detect these subtle cues, especially in tweets, due to limited context and a focus on surface patterns (Cheng et al., 2024). While Retrieval-Augmented Generation (RAG) aids in providing contexts, it does not always effectively uncover these nuanced political connections.

This study presents **GRASP-ChoQ**, a hybrid approach which improves LLM-based stance detection by combining knowledge graphs (KGs) and multistep reasoning. This approach addresses the dual challenges of adapting to dynamic political contexts and inferring indirect relational

[1] https://github.com/Programming-Dude/GRASP-ChoQ

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 8–30
December 23, 2025 ©2025 Association for Computational Linguistics

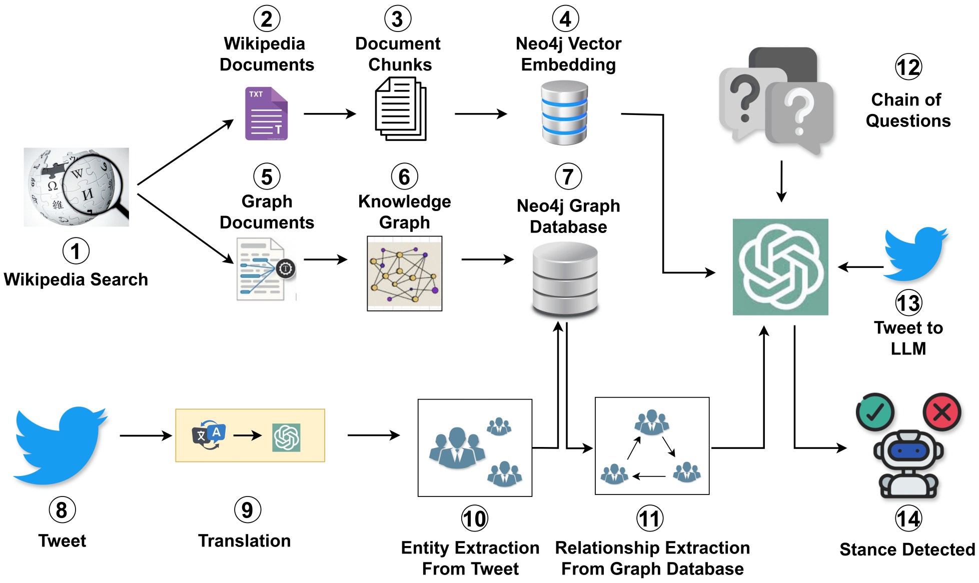


Figure 2: Our approach, GRASP-ChoQ, detects stance using two main components: Knowledge Graph and Chain-of-Questions Reasoning. The pipeline integrates Wikipedia search, a Neo4j graph database, and language models. Wikipedia documents are processed into chunks and stored in the knowledge graph. The graph database extracts entities and relationships, and a language model generates queries to determine the tweet's stance.

knowledge.

A significant aspect of the structure it contains is the incorporation of relational principles compared to **Social Balance Theory (SBT)** (Heider, 2013), which are frequently utilized to analyze harmony or stress in undirected triadic interactions (e.g., "a friend of my friend is my friend" or "a foe of my foe is my friend").

In order to describe asymmetric interactions using signed, directional edges, recent extensions of SBT to directed graphs have been devised (Aref et al., 2020). According to the principles of SBT, knowledge graph integration may be very helpful in detecting latent linkages by organizing the data in a manner that makes it possible to uncover hidden connections based on interactions that already exist (Chepurova et al., 2023). That is why our approach integrates this.

Approaching stance detection as a one-step decision limits the ability to reason through complex, indirect relationships. We address this by providing a set of questions (Zhu et al., 2023) that divides the process into smaller, more controllable phases, therefore facilitating the correct decision.

While prior works have separately explored knowledge-graphs, retrieval augmentation (e.g., RAG with static subgraph retrieval) and Chain-of-Thought (CoT) or Chain-of-Questions prompting for enhanced reasoning, no existing approach integrates multi-hop KG traversal with question-guided reasoning for political stance detection.

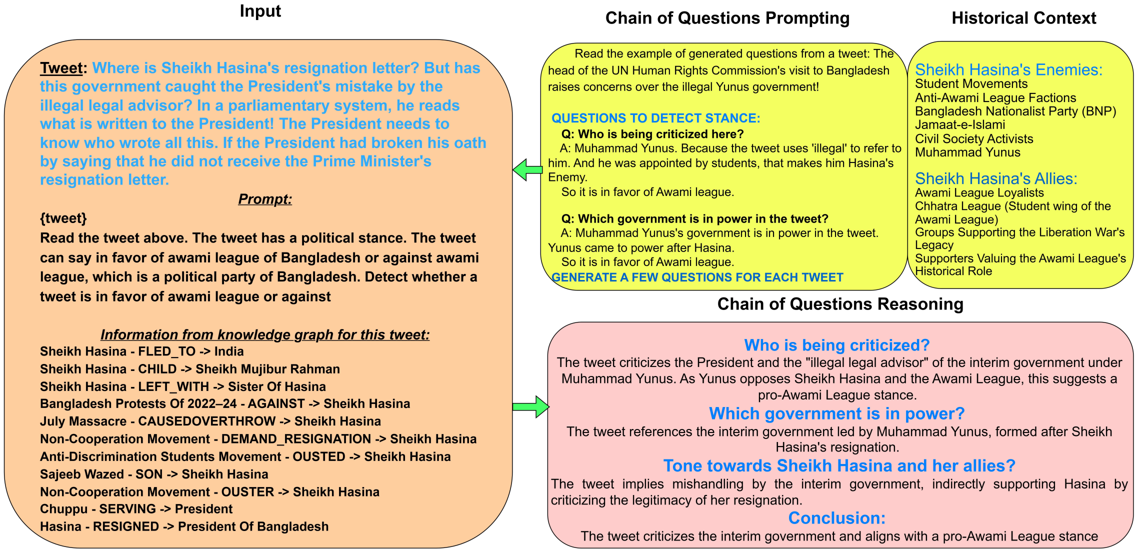
GRASP-ChoQ is particularly effective for complex political environments such as Bangladesh (Chowdury, 2024), a geopolitically significant region in recent times that remains underexplored in the context of target-based political stance detection (Bestvater and Monroe, 2023).

The main contributions of our research paper are as follows:

- We introduce Knowledge **G**raph-Based **R**etrieval **A**ugmentation for **S**tance Detection in **P**olitical Texts with **Ch**ain-**o**f-**Q**uestions Reasoning, shortly called **GRASP-ChoQ**. This approach combines two essential components: Knowledge Graph (KG) and Chain-of-Questions (ChoQ) to detect political stances effectively.

- We present **BPDisC** (**B**angladesh **P**olitical **Dis**course **C**orpus)-a novel dataset comprising tweets that explicitly express political stances related to the political climate in Bangladesh during and following the July 2024 protests.

- Our proposed method substantially improves the **reasoning capabilities** of baseline LLMs



Input

Tweet: Where is Sheikh Hasina's resignation letter? But has this government caught the President's mistake by the illegal legal advisor? In a parliamentary system, he reads what is written to the President! The President needs to know who wrote all this. If the President had broken his oath by saying that he did not receive the Prime Minister's resignation letter.

Prompt:

{tweet}
Read the tweet above. The tweet has a political stance. The tweet can say in favor of awami league of Bangladesh or against awami league, which is a political party of Bangladesh. Detect whether a tweet is in favor of awami league or against

Information from knowledge graph for this tweet:

Sheikh Hasina - FLED_TO -> India
Sheikh Hasina - CHILD -> Sheikh Mujibur Rahman
Sheikh Hasina - LEFT_WITH -> Sister Of Hasina
Bangladesh Protests Of 2022–24 - AGAINST -> Sheikh Hasina
July Massacre - CAUSEDOVERTHROW -> Sheikh Hasina
Non-Cooperation Movement - DEMAND_RESIGNATION -> Sheikh Hasina
Anti-Discrimination Students Movement - OUSTED -> Sheikh Hasina
Sajeeb Wazed - SON -> Sheikh Hasina
Non-Cooperation Movement - OUSTER -> Sheikh Hasina
Chuppu - SERVING -> President
Hasina - RESIGNED -> President Of Bangladesh

Chain of Questions Prompting

Read the example of generated questions from a tweet: The head of the UN Human Rights Commission's visit to Bangladesh raises concerns over the illegal Yunus government!

QUESTIONS TO DETECT STANCE:

Q: Who is being criticized here?
A: Muhammad Yunus. Because the tweet uses 'illegal' to refer to him. And he was appointed by students, that makes him Hasina's Enemy.
So it is in favor of Awami league.

Q: Which government is in power in the tweet?
A: Muhammad Yunus's government is in power in the tweet. Yunus came to power after Hasina.
So it is in favor of Awami league.

GENERATE A FEW QUESTIONS FOR EACH TWEET

Historical Context

Sheikh Hasina's Enemies:
Student Movements
Anti-Awami League Factions
Bangladesh Nationalist Party (BNP)
Jamaat-e-Islami
Civil Society Activists
Muhammad Yunus

Sheikh Hasina's Allies:
Awami League Loyalists
Chhatra League (Student wing of the Awami League)
Groups Supporting the Liberation War's Legacy
Supporters Valuing the Awami League's Historical Role

Chain of Questions Reasoning

Who is being criticized?
The tweet criticizes the President and the "illegal legal advisor" of the interim government under Muhammad Yunus. As Yunus opposes Sheikh Hasina and the Awami League, this suggests a pro-Awami League stance.

Which government is in power?
The tweet references the interim government led by Muhammad Yunus, formed after Sheikh Hasina's resignation.

Tone towards Sheikh Hasina and her allies?
The tweet implies mishandling by the interim government, indirectly supporting Hasina by criticizing the legitimacy of her resignation.

Conclusion:
The tweet criticizes the interim government and aligns with a pro-Awami League stance

Figure 3: This figure illustrates a structured approach which integrates information from knowledge graph, chain-of-questions prompting, historical context, and reasoning to classify tweets

in detecting political stances by addressing the dual challenges of adapting to dynamic political contexts and inferring indirect relational knowledge, as confirmed through an ablation study.

2 Related Works

Kopacheva et al. (2023) explored the structural roles of social networks in predicting the participation of protests in the context of ecological protests in Russia. The research emphasized how personal social networks predict political participation by highlighting network density and size as major attributes affecting the likelihood of participation in a protest action.

Panda et al. (2024) proposed HOLMES, a distilled knowledge graph approach for multi-hop QA, reducing token use and improving Exact Match on HotpotQA and MuSiQue, though challenged by graph incompleteness. Chen et al. (2024b) introduced a three-stage LLM-based pipeline for KGR without fine-tuning, enhancing KG enrichment and entity reranking, but limited by cost and scalability. Edge et al. (2024) presented Graph RAG for query-focused summarization via graph-based retrieval and indexing, requiring broader evaluation. Wu et al. (2024) introduced MedGraphRAG, a medical RAG model using triple-linked graphs and U-Retrievals, needing clinical validation and real-time updates. Garg and Caragea (2024) addressed target prioritization failures in stance detection with Stanceformer, a target-aware transformer that boosts self-attention scores, though constrained by LLM size and resources. Weinzierl and Harabagiu (2024) proposed "Tree-of-Counterfactual Prompting" (TR-ZSSD), a zero-shot stance detection method reliant on culturally informed models, limiting application in diverse settings. Chen et al. (2024a) enhanced FSA on complex financial texts using a four-step Chain-of-Thought LLM pipeline, showing top performance. Mei et al. (2024) proposed a contrastive learning approach for hateful meme detection, with an updatable embedding space that avoids retraining.

Shui et al. (2024) proposed a Llama3-8b-based emotion text classification model enhanced with LoRA and FlashAttention. Focused on stance detection in political tweets, it leverages knowledge graph principles for improved classification of complex opinions. However, dataset limitations restricted sentiment intensity regression to just two datasets, and hardware constraints prevented fine-tuning on larger models like Llama 3-70B, limiting results to 8B and 7B variants. Sreekala et al. (2024) introduced a news classification scheme using hierarchical clustering with ensemble methods. Documents were clustered with various aggregative techniques and classified using Gradient Boosting, Bagging, and Random Forests. A key limitation is its reliance on the BBC News dataset, which

Datasets	Target(s)	Source	Size	Time	Language	Annotation Classes
SemEval-2016 Task 6	Atheism, Climate Change is Concern, Feminist Movement, Hillary Clinton, Legalization of Abortion, Donald Trump	Twitter	4,870 tweets	2016	English	Favor, Against, Neither
P-stance	Donald Trump, Joe Biden, Bernie Sanders	Twitter	21,574 tweets	2020	English	Favor, Against, Neither
Mawqif	COVID-19 vaccine, digital transformation, women empowerment	Twitter	4,121 tweets	2022	Arabic	Favor, Against, None
Swami et al. (2018)	Demonetisation in India in 2016	Twitter	3,545 tweets	2018	English-Hindi	Favor, Against, None
Lai et al. (2018).	2016 referendum on reform of the Italian Constitution	Twitter	993 triplets (2,889 tweets)	2018	Italian	Favor, Against, None
Lüüsi et al. (2024)	Immigration-related sentences from Estonian news articles from Ekspress Grupp and Uued Uudised	News media	3,261 sentences	2015–2022	Estonian	Supportive, Neutral, Against
BPDisC (Ours)	**Awami League (Bangladeshi Political Party)**	**Twitter**	**2,847 tweets**	**2024**	**Bangla-English**	**Favor, Against**

Table 1: Stance detection datasets for the experiments. The datasets cover political topics across different languages, including English (Mohammad et al., 2016a; Li et al., 2021), Arabic (Alturayeif et al., 2022), Hindi, Italian, Estonian. For our experiments we only used FAVOR and AGAINST classes.

may not fully represent diverse news sources, affecting classification performance. Shimin (2024) explored book content classification using LLaMA's self-attention and word embedding mechanisms, demonstrating improved efficiency and accuracy in book classification.

3 Methodology

3.1 Retrieval Augmented Generation

RAG retrieves documents by measuring semantic similarity—typically via vector embeddings and metrics like cosine similarity—treating each document as an isolated point in embedding space (Barnett et al., 2024). This, however, ignores relational context such as citations or knowledge graph links.

For instance, in a citation network (Clough et al., 2015), papers P_1, P_2, P_3 may be connected via citation edges $P_1 \to P_2$ and $P_2 \to P_3$. If the logical citation path (*Q* → *P1* → *P2* → *P3*) is more informative, a retrieval system depending only on embeddings would prioritize retrieving *P3* if it has a high embedding similarity to a given query Q.

3.2 Knowledge Graph

Graph-based structures link pieces of information into nodes and edges, making it easy to organize and find what is needed (Xie et al., 2024). Just the relevant subgraph is then pulled from the knowledge graph to show how entities connect (Li et al., 2024). This turns long passages into a small, clear graph (Dong et al., 2024).

From Table 2 and Table 3, we can see how a long text is summarized and the key relationships are extracted. Next, we performed subgraph retrieval from the larger knowledge graph by identifying key entities within the tweet with LLM. This process involved extracting relevant entities and formulating a graph query to retrieve the most contextually appropriate subgraph.

Let:

- q be the input tweet.
- $E(q)$ represent the set of entities extracted from q.
- $g(e)$ represent the function that generates a query from entity e.
- $G(q)$ represent the result of querying the graph with query q.

The function can be expressed as:

$$\text{retriever}(q) = \bigcup_{e_i \in E(q)} G(g(e_i))$$

3.3 Chain of Questions

The primary motivation for adopting a Chain-of-Questions approach is to enable systematic decomposition into sub-questions. (Dua et al., 2022). By decomposing complex questions into simpler sub-questions, the model can focus on answering each sub-component accurately, thereby reducing the accumulation of errors that often occur in single-steps. As shown in Figure 3, the Chain-of-Questions prompting section features two open-ended questions (Ling et al., 2023) along with their corresponding answers, while another question is posed without an answer. In this scenario, the reasoning follows a clear and logical path.

3.4 Proposed Method

For each tweet, we extracted a relevant subgraph of information and generated a chain of questions through initial prompting designed to infer the stance of the tweet based on its contextual associations.

Figure 2 describes the overall workflow and association between modules and Figure 3 how the context is provided to LLM.

Let $\mathbf{D} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$ be the dataset, where $\mathbf{r}_i$ represents the i-th row. The iteration over the dataset is represented as:

$$\forall i \in \{1, 2, \ldots, N\}, \quad y_i = f(t_i, r, g(S_i), U_i, M)$$

Where y_i represents the stance for the i-th row. t_i is the translated tweet for the i-th row. r means text with historical relationships which is constant across all rows. S_i denotes structured data retrieved from knowledge graph for the i-th row. U_i is unstructured chunks of texts for the i-th row. M represents the baseline LLM model used by f for prediction. $g(S_i)$ function converts S_i (structured data) into a string representation.

After the stance function is iterated over all rows:

$$Y = \{y_i \mid i \in \{1, 2, \ldots, N\}\}$$

Here, Y is the final collection of political stances. y_i is predicted stance for the i-th row. $i \in \{1, 2, \ldots, N\}$ iterates over all N rows in the dataset.

4 Experimental Results

This section presents the experimental outcomes evaluating the proposed methods through quantitative metrics and qualitative analysis, emphasizing their strengths and limitations.

4.1 Experimental Settings

The experiment setup describes the data used, how it was prepared, and the steps involved. A brief overview is given here, with more details in Section A of the Appendix.

Initial Data Processing We collected tweets and extracted key entities such as persons, locations, and organizations.

Build & Store Knowledge Graph Using these entities, we pulled information from Wikipedia, structured it into a Knowledge Graph, and stored it in Neo4j.

Baseline Performance We tested baseline language models on the dataset to obtain initial results.

Integrating Our Method For each tweet, we retrieved relevant subgraphs and combined them with chain-of-questions in the model for enhanced stance detection.

Other Datasets We repeated the process on BPDisC and six benchmark datasets, consistently outperforming baselines.

4.1.1 BPDisC Dataset

Data Collection The dataset covers large-scale political discourse on Twitter during the 2024 Bangladesh Protests, collected between July 1 and December 31, 2024. We gathered around 10,000 publicly available tweets by employing keyword searches, hashtags, regional settings, and specific timestamps. Each tweet in the dataset is accompanied by metadata, including tweet id, handle name, timestamps, replies, retweets, likes. Tweets were obtained through the X (Twitter) API, this methodology aligns with the approaches utilized in earlier research studies (Mohammad et al., 2016b) and (Hossny and Mitchell, 2018).

Pre-processing After collecting tweets, we applied preprocessing by excluding image-based posts, users without clear political affiliation, and tweets under five words. Following these exclusions, we refined the dataset to 2,847 tweets. The dataset contained tweets in both Bangla and English. For non-English text within this final dataset, we utilized machine translation via GPT-4o to ensure data clarity and uniform analysis.

Node ID	Type	Context
Ahidul Islam	Person	In 2021, seven children of freedom fighters, including Ahidul Islam, filed a writ petition challenging the quota system decision.
High Court	Organization	On 5 June 2024, High Court declared the decision to scrap the quota system invalid.
Students	Group	Key protest action staged by students during the intensified July demonstrations.
Police	Organization	Police used excessive force during protests, leading to clashes while attempting to quell student agitations.
Bangla Blockade	Event	Immediately after the verdict announcement, students protested in various universities. The movement intensified in July with blockades like "Bangla Blockade".

Table 2: Structured Extraction of Entities and Relationships from Wikipedia using LLM—The table represents extracted nodes (entities) and their contextual relationships from Wikipedia text.

Source Node	Relation	Target Node
Ahidul Islam	FILED	High Court
High Court	DECLARED	Quota System
Students	STAGED	Bangla Blockade
Police	USED_FORCE	Students

Table 3: Extracted Relationships using LLM Demonstrating Connections Between Key Entities.

Annotation Strategy Following prior strategies on inferring political affiliation from online profiles (Baran et al., 2022; Huszár et al., 2022; Johnson and Goldwasser, 2016), we manually annotated a subset of Twitter user accounts based on political cues found in their bios. Specifically, we identified users' stances in favor or against the Awami League, the most frequently referenced political party in our dataset. 32 Twitter accounts were labeled as in favor of the Awami League, while 17 accounts were labeled as against. These annotations were based on explicit expressions in user bios, such as slogans, hashtags, or mentions of party leaders. Accounts without such information were excluded, along with celebrity and news media outlets, as they generally maintain a neutral stance. Hence, we had to drop a massive portion of originally collected ~10k tweets.

Once annotated, the political stance label of each account was propagated to all tweets posted by that user in the dataset. This approach allowed us to construct a larger dataset of labeled tweets based on account-level annotations. In total, this yielded 1,512 tweets labeled as class 1 (in favor) and 1,335 tweets as class 0 (against), forming a final dataset suitable for supervised learning and analysis.

4.1.2 Inter-Annotator Agreement

We conducted a two-stage inter-annotator agreement process to ensure the reliability of the BPDisC dataset. This validation was performed first at the account level to confirm the user-level annotations and subsequently at the tweet level to validate the propagation of those labels. We used Cohen's Kappa(Cohen, 1960) to measure the consistency at both stages.

Account Level For the user-level annotations, two coders independently assessed 49 Twitter accounts (32 labeled as 'in favor' and 17 as 'against'), based on the explicit political cues in their bios. It was also confirmed that none of the 49 accounts were linked to recognized media outlets in Bangladesh which are generally regarded as politically neutral. The two coders reached an agreement on 47 out of the 49 accounts. To measure the consistency of this process, we calculated the inter-annotator agreement, achieving a **Cohen's Kappa of 0.91**, which indicates strong agreement and confirms the reliability of our account-level annotation.

Tweet Level To further validate the dataset's annotation, we also assessed the reliability of propagating account-level labels to individual tweets. A random subset of 300 tweets was independently annotated by two coders to determine if the tweet's content matched the account's propagated stance. The annotators agreed on 281 of the 300 tweets. This resulted in a **Cohen's Kappa of 0.87**. This strong level of agreement confirms that the stances expressed

Methods	LLMs	Accuracy	Precision	Recall	F1 Score
Zero-shot	GPT-4	58.00	80.04	27.85	41.30
	GPT-4o	59.61	84.81	29.17	43.30
	Gemini 1.5 Pro	59.64	75.17	35.85	48.54
	Mistral Large Latest	53.39	70.79	20.83	32.19
	DeepSeek R1	61.40	75.40	40.54	52.73
Few-shot	GPT-4	51.07	63.53	18.33	28.39
	GPT-4o	53.07	63.66	27.12	38.03
	Gemini 1.5 Pro	51.46	64.77	18.85	29.24
	Mistral Large Latest	50.09	60.76	17.00	26.56
	DeepSeek R1	62.59	81.26	38.43	52.18
Few Shot with Naive-RAG	GPT-4	56.41	78.60	24.54	37.50
	GPT-4o	63.96	83.11	40.34	54.32
	Gemini 1.5 Pro	57.50	81.29	25.86	39.24
	Mistral Large Latest	54.97	74.47	23.15	35.32
	DeepSeek R1	64.56	75.90	48.74	59.36
GRASP-ChoQ (Ours)	GPT-4	88.55	88.61	90.01	89.30
	GPT-4o	85.04	87.81	83.40	85.85
	Gemini 1.5 Pro	75.10	87.49	61.97	72.55
	Mistral Large Latest	72.60	83.83	59.99	69.93
	DeepSeek R1	93.12	96.14	90.67	**93.33**

Table 4: Comparison of different methods (Zero-shot, Few-shot, Naive-RAG, and GRASP-ChoQ) across models (Mistral, GPT-4, GPT-4o, Gemini 1.5 Pro, and DeepSeek R1) in terms of Accuracy, Precision, Recall, and F1 Score on **BPDisC** Dataset.

in individual tweets are highly consistent with the account-level labels, validating the reliability of the BPDisC dataset.

4.1.3 Other Datasets

In addition to our **BPDisC** dataset, we evaluated our approach on six other datasets.

For English-language, we utilized SemEval-2016 Task-6 and P-stance dataset (Mohammad et al., 2016a; Li et al., 2021). The dataset from (Swami et al., 2018) includes English–Hindi code-mixed tweets related to India's 2016 demonetization.

Beyond English, we employed non-English datasets: *Mawqif*, an Arabic-dialect Twitter corpus (Alturayeif et al., 2022); the dataset from (Lai et al., 2018), comprising Italian tweets related to the 2016 referendum on constitutional reform; and (Lüüsi et al., 2024), which features Estonian news media sentences discussing immigration. The datasets are detailed in Table 1.

We first removed stances labeled as *None*, *Neither*, or *Neutral* from the datasets. Next, we performed translation, entity extraction, and knowledge graph construction. Finally, we tailored prompts to each dataset and applied both zero-shot and GRASP-ChoQ methods.

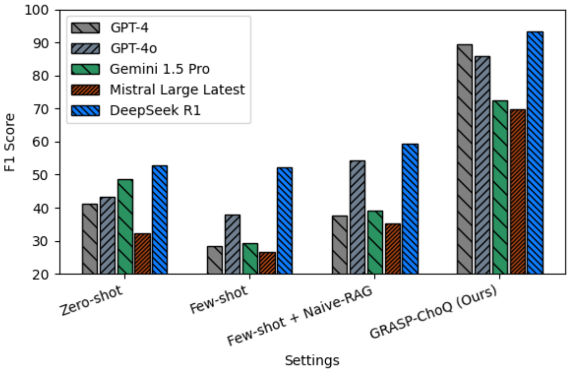


Figure 4: DeepSeek R1 Demonstrates Superior Performance Across All Input Settings.

4.1.4 Neo4j Configuration

Texts sourced from Wikipedia (Yu et al., 2021) were transformed into a graph database using Neo4j (Miller, 2013), a platform tailored for managing highly interconnected knowledge graphs. The database was represented as a graph comprising 883 nodes and 2,281 relationships for the BPDisC dataset.

In this graph, nodes represented key entities such

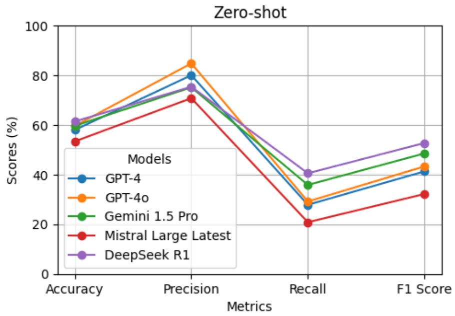


Figure 5: The performance of five AI models (GPT-4, GPT-4o, Gemini 1.5 Pro, Mistral Large Latest, and DeepSeek R1) across four evaluation metrics in a zero-shot setting.

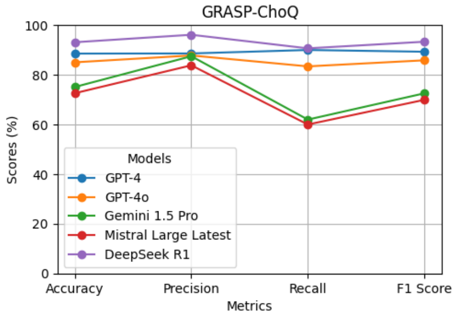


Figure 6: The performance of five AI models across four evaluation metrics for GRASP-ChoQ.

as users, topics, or hashtags, while relationships captured interactions.

Neo4j's Cypher query language enabled extraction of related subgraphs (Anuyah et al., 2024).

4.1.5 Baseline LLMs

Mistral Large Latest Developed by French startup Mistral AI (Jiang et al., 2023), this top-tier model builds on earlier versions with billions of parameters.

OpenAI GPT-4 The latest in the GPT series, GPT-4 surpasses GPT-3.5 in context handling, reasoning, and comprehension (OpenAI, 2023).

OpenAI GPT-4o An optimized GPT-4 variant offering faster, more accurate, and versatile performance (OpenAI, 2024).

Gemini 1.5 Pro DeepMind's multimodal model improves on Gemini 1.0 with longer context, faster processing, and enhanced performance (DeepMind, 2023).

DeepSeek R1 Launched in 2023 by DeepSeek AI, this open-source model is adopted by major firms like Microsoft and Amazon for its cost-effectiveness and accessibility(DeepSeekAI, 2025).

4.2 Ablation Study

We performed an ablation study by excluding two fundamental elements of GRASP-ChoQ:

- Retrieval and knowledge graph data integration into the reasoning process.
- ChoQ framework for breaking down intricate searches into sequential, linked inquiries.

Our analysis reveals that simple RAG is insufficient for complex political texts. While the DeepSeek-R1 baseline achieved a 52.73 F1 score, our full GRASP-ChoQ framework reached 93.33 F1. This massive +40.6 point improvement confirms the critical synergy between our components. The high performance is not attributable to just one part, but to the powerful combination of graph-based retrieval and structured reasoning.

Our ablation study confirms the components are complementary. From Table 5, we can clearly see the improvements of baseline models with both components.

Knowledge Graph provides superior, structured information between entities. Removing it untethered the model from factual evidence, causing it to rely on simple keywords rather than contextual relationships. The essential reasoning structure is supplied by Chain-of-Questions. When removed, the model lost its ability to perform multi-step logic, defaulting to single-turn predictions that failed to interpret subtle or indirect viewpoints.

These results prove that the model's success stems directly from the synergy between its parts: the Knowledge Graph provides the external understanding, while ChoQ enforces the disciplined reasoning required to use it. Both are essential for maintaining accuracy and interpretability.

4.3 Performance and Results

Initial evaluations on the BPDisC dataset (Table 4) showed that the DeepSeek-R1 model, enhanced by our proposed GRASP-ChoQ approach, achieved superior performance across all metrics compared to other models. Figure 4 and Figure 5 illustrate DeepSeek R1's strong baseline

Dataset	Model	Zero-shot	GRASP-ChoQ
BPDisC	GPT-4	41.30	89.30 *(+48.0)*
	GPT-4o	43.30	85.85 *(+42.5)*
	Gemini 1.5 Pro	48.54	72.55 *(+24.0)*
	Mistral Large Latest	32.19	69.93 *(+37.7)*
	DeepSeek R1	**52.73**	**93.33** *(+40.6)*
SemEval 2016	DeepSeek R1	95.07	96.81 *(+1.7)*
P-Stance	DeepSeek R1	97.61	98.33 *(+0.7)*
Mawqif	DeepSeek R1	84.37	95.63 *(+11.3)*
Swami et al.	DeepSeek R1	87.31	92.56 *(**+5.3**)*
Lai et al.	DeepSeek R1	74.21	90.83 *(+16.6)*
Lüüsi et al.	DeepSeek R1	79.35	92.94 *(+13.6)*

Table 5: F1 Score Comparison of Models on Various Datasets in Zero-shot and GRASP-ChoQ methods

performance across various settings. Based on this finding, we proceeded to evaluate only the DeepSeek-R1 model on the remaining six datasets. GRASP-ChoQ significantly enhanced DeepSeek R1's performance on these datasets as well. Table 5 highlights this improvement.

Understanding Performance Gains: From Table 5, we can see that English language datasets have strong baseline performance, our approach increases F1 scores slightly. This happens because the target entities of these datasets are well-known and LLMs have their information in the pre-training data already. But for other language datasets, where target entities are not so well known, our approach improves DeepSeek-R1's baseline performance.

So, from the experiments and results, it can be observed that, our approach is beneficial for low-resource political texts whose target entities are not well-studied.

5 Conclusion

Large Language Models (LLMs) often demonstrate suboptimal performance when tasked with processing low-resource political information. To date, no prior research has attempted to classify political stances in the context of Bangladesh, presenting a unique opportunity for exploration. Our proposed approach GRASP-ChoQ, offers a novel strategy for addressing this gap. This approach was applied to the novel BPDisC and six other datasets. By fostering critical thinking, this method has the potential to be adapted to other political contexts. This can be achieved by constructing a tailored knowledge base and designing a carefully curated chain of questions specific to each contextual framework.

Limitations

The GRASP-ChoQ method has several limitations. It relies solely on textual data, excluding multimodal inputs like images or videos, which are increasingly important in political communication these days. This restricts the method's ability to capture visual contexts. Moreover, the exclusion of image data means the final dataset is smaller than it could have been. Additionally, our main focus was to reduce the cost of fine-tuning large models, so no additional training was performed, which may limit adaptability. Furthermore, the limited knowledge base may restrict the approach's ability to generalize across diverse political contexts. Accurately identifying political sarcasm—a subtle yet pervasive feature of political discourse—also remains a significant challenge. While the inter-annotator agreement process demonstrated strong reliability at both the account and tweet levels, limitations exist in terms of the small sample sizes. The focus on accounts without recognized media links may not fully capture the political diversity present in the broader Twitter ecosystem.

Ethics and Bias Mitigation

This paper guarantees openness, fairness, and privacy protection by following strict ethical criteria in stance identification for political debate. All the information used in this study came from publicly accessible sources, including X (Twitter). The final dataset was relatively balanced. This distribution reduces the risk of systematic bias in classification outcomes based on label imbalance. However, large language models (LLMs) used in this study may still reflect biases inherited during pretraining on broad internet data.

We ensured that both Pro-Awami League and Anti-Awami League classifications were treated equitably. We have meticulously documented all data preprocessing steps and KG made to facilitate independent verification. We conducted this research solely for academic and research purposes, with a commitment to upholding the ethical standards established for public data analysis. No aspect of this research is intended to be used for political profiling, influence campaigns, or any activities that may lead to the manipulation or misrepresentation of political discourse. We acknowledge the ethical implications of automated stance detection in sensitive political contexts and emphasize the responsible use of such technologies.

References

Nora Saleh Alturayeif, Hamzah Abdullah Luqman, and Moataz Aly Kamaleldin Ahmed. 2022. Mawqif: A multi-label arabic dataset for target-specific stance detection. In *Proceedings of the Seventh Arabic Natural Language Processing Workshop (WANLP)*, pages 174–184.

Sydney Anuyah, Victor Bolade, and Oluwatosin Agbaakin. 2024. Understanding graph databases: a comprehensive tutorial and survey. *arXiv preprint arXiv:2411.09999*.

Samin Aref, Ly Dinh, Rezvaneh Rezapour, and Jana Diesner. 2020. Multilevel structural evaluation of signed directed social networks based on balance theory. *Scientific reports*, 10(1):15228.

Joanna Baran, Michał Kajstura, Maciej Ziółkowski, and Krzysztof Rajda. 2022. Does twitter know your political views? politweets dataset and semi-automatic method for political leaning discovery. *arXiv preprint arXiv:2207.07586*.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 194–199.

Samuel E Bestvater and Burt L Monroe. 2023. Sentiment is not stance: Target-aware opinion classification for political text analysis. *Political Analysis*, 31(2):235–256.

Tianyu Chen, Yiming Zhang, Guoxin Yu, Dapeng Zhang, Li Zeng, Qing He, and Xiang Ao. 2024a. EFSA: Towards event-level financial sentiment analysis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7455–7467. Association for Computational Linguistics.

Zhongwu Chen, Long Bai, Zixuan Li, Zhen Huang, Xiaolong Jin, and Yong Dou. 2024b. A new pipeline for knowledge graph reasoning enhanced by large language models without fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1381, Miami, Florida, USA. Association for Computational Linguistics.

Ning Cheng, Zhaohui Yan, Ziming Wang, Zhijie Li, Jiaming Yu, Zilong Zheng, Kewei Tu, Jinan Xu, and Wenjuan Han. 2024. Potential and limitations of llms in capturing structured semantics: A case study on srl. In *International Conference on Intelligent Computing*, pages 50–61. Springer.

Alla Chepurova, Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2023. Better together: Enhancing generative knowledge graph completion with language models and neighborhood information. *arXiv preprint arXiv:2311.01326*.

Saeyd Rashed Hasan Chowdury. 2024. The role of political parties in bangladesh's july revolution of 2024: Insights from sufi perspectives. *International Journal of Research and Innovation in Social Science*, 8(11):2077–2093.

James R Clough, Jamie Gollings, Tamar V Loach, and Tim S Evans. 2015. Transitive reduction of citation networks. *Journal of Complex Networks*, 3(2):189–203.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Google DeepMind. 2023. Gemini 1.5: Unlocking multimodal understanding across millions of tokens.

DeepSeekAI. 2025. Deepseek-r1: Incentivizing reasoning capability in large language models via reinforcement learning.

Jialin Dong, Bahare Fatemi, Bryan Perozzi, Lin F Yang, and Anton Tsitsulin. 2024. Don't forget to connect! improving rag with graph-based reranking. *arXiv preprint arXiv:2405.18414*.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Karan Garg and Cornelia Caragea. 2024. Stanceformer: Target-aware transformer for stance detection. *arXiv preprint*, arXiv:2410.07083.

Fritz Heider. 2013. *The psychology of interpersonal relations*. Psychology Press.

Ahmad Hany Hossny and Lewis Mitchell. 2018. Event detection in twitter: A keyword volume approach. In *2018 IEEE international conference on data mining workshops (ICDMW)*, pages 1200–1208. IEEE.

Ferenc Huszár, Sofia Ira Ktena, Conor O'Brien, Luca Belli, Andrew Schlaikjer, and Moritz Hardt. 2022. Algorithmic amplification of politics on twitter. *Proceedings of the national academy of sciences*, 119(1):e2025334119.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Kristen Johnson and Dan Goldwasser. 2016. Identifying stance by analyzing political discourse on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 66–75.

Elizaveta Kopacheva, Masoud Fatemi, and Kostiantyn Kucher. 2023. Using social-media-network ties for predicting intended protest participation in russia. *OsnEM*, n/a:n/a.

Mirko Lai, Viviana Patti, Giancarlo Ruffo, and Paolo Rosso. 2018. Stance evolution and twitter interactions in an italian political debate. In *Natural Language Processing and Information Systems: 23rd International Conference on Applications of Natural Language to Information Systems, NLDB 2018, Paris, France, June 13-15, 2018, Proceedings 23*, pages 15–27. Springer.

Philip Leifeld and Laurence Brandenberger. 2019. Endogenous coalition formation in policy debates. *arXiv preprint arXiv:1904.05327*.

Y. Li, R. Zhang, and J. Liu. 2024. An enhanced prompt-based llm reasoning scheme via knowledge graph-integrated collaboration. In *International Conference on Artificial Neural Networks*, pages 251–265, Cham. Springer Nature Switzerland.

Yingjie Li, Tiberiu Sosea, Aditya Sawant, Ajith Jayaraman Nair, Diana Inkpen, and Cornelia Caragea. 2021. P-stance: A large dataset for stance detection in political domain. In *Findings of the association for computational linguistics: ACL-IJCNLP 2021*, pages 2355–2365.

Chen Ling, Xuchao Zhang, Xujiang Zhao, Yanchi Liu, Wei Cheng, Mika Oishi, Takao Osaki, Katsushi Matsuda, Haifeng Chen, and Liang Zhao. 2023. Open-ended commonsense reasoning with unrestricted answer candidates. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8035–8047.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. 2024. Datasets for large language models: A comprehensive survey. *arXiv preprint arXiv:2402.18041*.

Yujian Liu, Xinliang Frederick Zhang, David Wegsman, Nick Beauchamp, and Lu Wang. 2022. Politics: Pretraining with same-story article comparison for ideology prediction and stance detection. *arXiv preprint arXiv:2205.00619*.

Lauri Lüüsi, Uku Kangur, Roshni Chakraborty, and Rajesh Sharma. 2024. Political stance detection in estonian news media. In *Proceedings of the 9th International Workshop on Computational Linguistics for Uralic Languages*, pages 12–28.

Jianzong Mei, Jing Chen, Wei Lin, Bill Byrne, and Marcus Tomalin. 2024. Improving hateful meme detection through retrieval-guided contrastive learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5333–5347. Association for Computational Linguistics.

Justin J Miller. 2013. Graph database applications and concepts with neo4j. In *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*, volume 2324, pages 141–147.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016a. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016b. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 31–41.

OpenAI. 2023. Gpt-4 technical report.

OpenAI. 2024. Gpt-4o system card.

Michael T Oswald, Meike Fromm, and Elena Broda. 2021. Strategic clustering in right-wing-populism¿green policies' in germany and france. *Zeitschrift für Vergleichende Politikwissenschaft*, 15(2):185–205.

Pranoy Panda, Ankush Agarwal, Chaitanya Devaguptapu, Manohar Kaul, and Prathosh A P. 2024. Holmes: Hyper-relational knowledge graphs for multi-hop question answering using llms. In *Proceedings of the 2024 Annual Meeting of the Association for Computational Linguistics (ACL 2024)*.

David Rozado. 2024. The political preferences of llms. *arXiv preprint arXiv:2402.01789*.

F. Shimin. 2024. Application research on large language model attention mechanism in automatic classification of book content. In *2024 IEEE 2nd International Conference on Image Processing and Computer Applications (ICIPCA)*, pages 343–350, Shenyang, China. IEEE.

H. Shui, Y. Zhu, F. Zhuo, Y. Sun, and D. Li. 2024. An emotion text classification model based on llama3-8b using lora technique. In *2024 7th International Conference on Computer Information Science and Application Technology (CISAT)*, pages 380–383, Hangzhou, China. IEEE.

K. Sreekala, M. Sivajyothi, D. Chiranjevi, G. Sunitha, and M. Swetha. 2024. A novel integration of hierarchical clustering and ensemble classification algorithms for news classification. In *2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC - ROBINS)*, pages 524–528, Coimbatore, India. IEEE.

Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. An english-hindi code-mixed corpus: Stance annotation and baseline system. *arXiv preprint arXiv:1805.11868*.

The Economist. 2024. The economist's country of the year for 2024. *The Economist*.

Michael Weinzierl and Sanda Harabagiu. 2024. Tree-of-counterfactual prompting for zero-shot stance detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–880. Association for Computational Linguistics.

J. Wu, J. Zhu, Y. Qi, J. Chen, M. Xu, F. Menolascina, and V. Grau. 2024. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. *arXiv preprint arXiv:2408.04187*.

Weijian Xie, Xuefeng Liang, Yuhui Liu, Kaihua Ni, Hong Cheng, and Zetian Hu. 2024. Weknow-rag: An adaptive approach for retrieval-augmented generation integrating web search and knowledge graphs. In *Proceedings of the KDD Cup CRAG Workshop 2024*, Barcelona, Spain. ACM.

Haoze Yu, Haisheng Li, Dianhui Mao, and Qiang Cai. 2021. A domain knowledge graph construction method based on wikipedia. *Journal of Information Science*, 47(6):783–793.

Wang Zhu, Jesse Thomason, and Robin Jia. 2023. Chain-of-questions training with latent answers for robust multistep question answering. *arXiv preprint arXiv:2305.14901*.

Appendix

A Experimental Setup

All prompting experiments were conducted using GPT-4, GPT-4o, Gemini 1.5 Pro, DeepSeek R1 and Mistral Large (latest), with a temperature setting of 0 and all other parameters maintained at their default values to ensure consistency and reproducibility. We used the OpenAI SDK for interfacing with all models, thereby streamlining the experimental process and ensuring a uniform execution environment. Consistent hyperparameters were applied across the board, ensuring that any performance differences could be confidently attributed to the intrinsic capabilities of each model rather than to variations in the configuration or interfacing method.

A.1 Dataset Details

Group	Distinct Handles
Against	@ZulkarnainSaer, @MushfiqulFazal, @mrforayeji, @redwanxyz, @theBDarmy, @tasneem, @UNinIndia, @support_yunus, @shafiqalam2024, @EUinBangladesh, @BNPBdMediaCell, @volker_turk, @muktadirnewage, @Oliver_Tomarket, @JonFDanilowicz, @ChiefAdviserGoB, @dhruvrahtee
Favor	@alliance29464, @sajeebwazed, @Chellaney, @albd1971, @bdperspectives, @ATeam_1971, @bdwatch2024, @k_shayera, @VNouka, @NeenaRai, @Asifurrahman71, @pressxpresspx, @sumon_tarek, @MAarafat71, @BD_DiGEST, @AsadZam89687147, @RUSI_org, @CJBdingo25, @istiak_ahmmad, @Udashi_Pothik, @MaryMillben, @OnceAgainHasina, @SushantaDGupta, @INSIGHTUK2, @amnestysasia, @IndiaToday, @iindrojit, @Bangladesh_Fact, @TimesAlgebraIND, @CChoddogram, @FreedomRightsRL, @sagor250

Table 6: Distinct Twitter handles grouped by stance

A.2 Tweet Translation Prompt

A.2.1 For BPDisC Dataset

For translating tweets from Bangla to English, preserving proper nouns (like names of people and organizations), and leaving English tweets unchanged, we use the following prompt structure. Few-shot examples are included to guide the Large Language Model (LLM).

LOC	Count	PER	Count	ORG	Count
bangladesh	2661	yunus	991	government	921
country	484	hasina	717	league	873
dhaka	276	sheikh	608	awami	731
india	242	rahman	135	police	441
bangladeshi	183	bangabandhu	128	bnp	366
state	145	muhammad	122	army	309
bangladeshcrisis	133	asif	108	jamaat	197
uk	116	nazrul	101	chhatra	148

Table 7: Top named entities by type with mention frequency

Task: Translate the following tweet from Bangla to English. If the tweet is already in English, output the original tweet. Do NOT translate proper nouns (e.g., names of people, organizations, specific places).

Examples: Use the following examples to understand the task better.

Tweet (Bangla): প্রধানমন্ত্রী শেখ হাসিনা আজ একটি নতুন প্রকল্পের উদ্বোধন করেছেন।
Translated Tweet (English): Prime Minister Sheikh Hasina inaugurated a new project today.
Tweet (English): Just attended the Google I/O conference.
Translated Tweet (English): Just attended the Google I/O conference.

Tweet (Bangla): BRAC বিশ্বের অন্যতম বৃহত্তম এনজিও।
Translated Tweet (English): BRAC is one of the world's largest NGOs.

A.2.2 For Other Datasets

To support multilingual datasets, we use the following prompt structure. The Language Model (LLM) is instructed to translate non-English tweets into English, while preserving proper nouns (e.g., names of people, organizations, or specific places), and to leave tweets already in English unchanged. Few-shot examples are included to guide the model's behavior.

bangladesh
interim government
india
dhaka
right
group
human right
prime minister
league leader
bank
terrorist
first
three
must
family
reform
way
good
role
movement
hasina
death
media
time
day
member
august
indian
one
sheikh
hasina
student
attack
yunus
new
bnp
awami
league
many
report
government
police
protest
army
power
leader
year
today
country
people
party
two
violence
political
even
case
district
justice

Figure 7: Most frequent entities from the collected tweets

Task: Translate the following tweet into English. If the tweet is already in English, output the original tweet. Do **NOT** translate proper nouns (e.g., names of people, organizations, specific places).

Examples: Use the following examples to understand the task better.

Tweet (Non-English): [Example tweet in a non-English language with proper nouns]
Translated Tweet (English): [Correct English translation, preserving proper nouns]

Tweet (English): [Example English tweet]
Translated Tweet (English): [Same English tweet]

Tweet (Non-English): [Another example tweet in a different non-English language]
Translated Tweet (English): [Correct English translation, preserving proper nouns]

A.3 Graph Construction from Supporting Documents

To build a knowledge graph from unstructured text, we use Wikipedia as a source of supporting documents. We retrieve relevant content based on a user query, split the documents into smaller chunks, convert them into graph-structured representations using a language model, and finally populate the graph. This structured format helps capture both entities and relationships for downstream reasoning.

Step 1: Load Wikipedia content using a query.

```
from langchain.document_loaders
import WikipediaLoader

raw_documents = WikipediaLoader(
query="bangladesh student uprising
of july 2024"
).load()
```

Step 2: Split documents into token-sized chunks.

```
from langchain.text_splitter
import TokenTextSplitter

text_splitter = TokenTextSplitter(
chunk_size=256, chunk_overlap=50)
documents = text_splitter.
split_documents(
raw_documents)
```

Step 3: Convert document chunks into graph-structured format.

```
from langchain_experimental.
graph_transformers
import LLMGraphTransformer
llm_transformer =
LLMGraphTransformer(llm=llm)

graph_documents = llm_transformer.
convert_to_graph_documents(
[raw_documents])
```

Step 4: Add graph documents to the knowledge graph.

```
graph.add_graph_documents(
    graph_documents,
    baseEntityLabel=True,
    include_source=True
)
```

A.4 Hybrid Vector Indexing and Full-Text Search

To support both dense and keyword-based retrieval, we create a hybrid vector index using sentence embeddings from a pre-trained model. These embeddings are stored in a Neo4j-backed vector store. Additionally, a full-text index is created for entity-level lookup to complement semantic search with symbolic filtering.

Step 1: Define the embedding model.

```
from langchain.embeddings import
HuggingFaceEmbeddings
embeddings = HuggingFaceEmbeddings(
model_name="sentence-transformers/
all-MiniLM-L6-v2"
)
```

Step 2: Create a hybrid vector index from the graph.

```
from langchain.vectorstores import
Neo4jVector
vector_index                    =
Neo4jVector.from_existing_graph(
embeddings,
search_type="hybrid",
node_label="Document",
text_node_properties=["text"],
embedding_node_property="embedding"
)
```

Step 3: Create a full-text index on entity nodes.

```
graph.query("CREATE FULLTEXT INDEX
entity IF NOT EXISTS
FOR (e:__Entity__) ON EACH [e.id]")
```

A.5 Entity Extraction and Full-Text Query Generation

To extract entities (person, organization, business) from the input text and generate full-text queries for a Neo4j-based search, we use the following approach.

Step 1: Define the entity extraction model.

```
from langchain_core.pydantic_v1
import BaseModel, Field
from typing import List
class Entities(BaseModel):
# Identifying information about
entities.
names: List[str] = Field(...,
description="All the person,
organization, location entities
that appear in the text")
```

Step 2: Set up the prompt and model chain for extraction.

```
from langchain_core.prompts import
ChatPromptTemplate
prompt                              =
ChatPromptTemplate.from_messages([
("system",  "You  are  extracting
location, organization and person
entities from the text."),
("human",  "Use  the  given  format
to  extract  information  from  the
following input: question")
])
entity_chain     =     prompt     |
llm.with_structured_output(Entities)
```

Step 3: Define the full-text query generation function.

```
from langchain_community.
vectorstores.neo4j_vector   import
remove_lucene_chars
def generate_full_text_query(input:
str) -> str:
full_text_query = ""
words    =    [el    for    el    in
remove_lucene_chars(input).split()
if el]
for word in words[:-1]:
full_text_query += f" word 2 AND"
full_text_query += f" words[-1] 2"
return full_text_query.strip()
```

Step 4: Implement structured retrieval using entity names.

```
def  structured_retriever(question:
str) -> str:
time.sleep(0.5)
result = ""
entities                           =
entity_chain.invoke("question":
question)
for entity in entities.names:
response = graph.query(
"CALL db.index.fulltext.queryNodes(
'entity', $query, limit:2)"
YIELD node,score
CALL {
WITH node
MATCH (node)-[r:!MENTIONS]->(neighbor)
RETURN node.id + ' - ' + type(r) +
' -> ' + neighbor.id AS output
UNION ALL
WITH node
MATCH (node)<-[r:!MENTIONS]-(neighbor)
RETURN neighbor.id + ' - ' + type(r)
+ ' -> ' + node.id AS output

RETURN output LIMIT 50
)
result += "
n".join([el['output']  for  el  in
response])
return result
```

B Baseline & GRASP-ChoQ Prompts

B.1 Prompt used in zero-shot

Target Entity: Awami League
Task: Read the tweet and determine whether it expresses a stance **in FAVOR of** or **AGAINST** the specified target entity.

Tweet: [tweet]

B.2 Prompt used in few-shot

Target Entity: Awami League
Task: Analyze the following tweets and determine if the author's stance is **in FAVOR of** or **AGAINST** the specified target entity.

Tweet: "The country is moving forward under the leadership of Sheikh Hasina. #AwamiLeague"
Stance: Favor

Tweet: "Corruption is rampant, and the government is not listening to the people. #Bangladesh"
Stance: Against

Tweet: "{New tweet text goes here}"
Stance:

B.3 Prompt used in few-shot with Naive-RAG

Target Entity: Awami League
Context: [retrieved context]
Task: Analyze the following tweets and determine if the author's stance

is **in FAVOR of** or **AGAINST** the specified target entity, using the tweet and context provided.

Tweet: "The country is moving forward under the leadership of Sheikh Hasina. #AwamiLeague"
Context: "Sheikh Hasina is the leader of the Awami League and has been praised for infrastructure development."
Stance: Favor

Tweet: "Corruption is rampant, and the government is not listening to the people. #Bangladesh"
Context: "The Awami League has been criticized in the media for alleged corruption and authoritarian practices."
Stance: Against

Tweet: "{New tweet text goes here}"
Context: "{Retrieved context goes here}"
Stance:

B.4 Prompts Used in GRASP-ChoQ For BPDisC Dataset

We design prompts to evaluate whether a given tweet exhibits a political stance **in favor of** or **against** the **Awami League**, a major political party in Bangladesh. The prompt integrates contextual reasoning and guided questioning to emulate how annotators or language models might use external political knowledge to infer implicit stance.

Tweet: The head of the UN Human Rights Commission's visit to Bangladesh raises concerns over the illegal Yunus government!

Read the tweet above. The tweet has a political stance. It may express a view either **in favor of the Awami League of Bangladesh** or **against it**. Detect the stance of the tweet with respect to the Awami League. Use reasoning based on political references or implied affiliations.

ASK QUESTIONS TO DETECT STANCE:

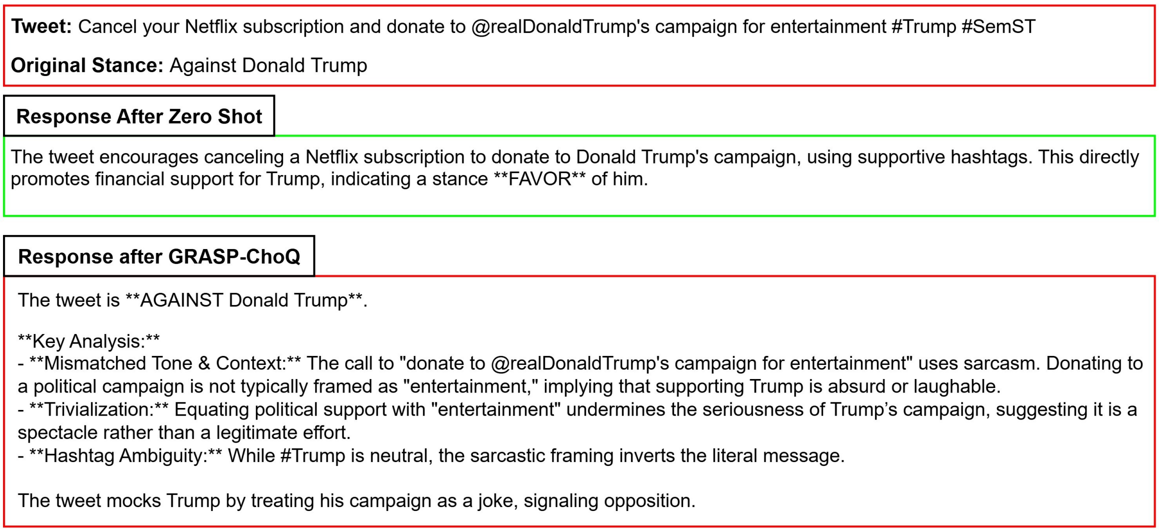
Q: Who is being criticized here?
A: Muhammad Yunus. Because the tweet uses "illegal" to describe him. Since Yunus is opposed to Sheikh Hasina (leader of Awami League), this implies support for Awami League.
Q: Which government is depicted in power in the tweet? A: Muhammad Yunus's government. As he is seen to follow Hasina, and is portrayed negatively, the stance favors the Awami League.

To aid your decision, general background knowledge about political figures and affiliations is provided.

INFO_FROM_KNOWLEDGE_GRAPH:
Sheikh_Hasina - LEADER -> Awami_League
Sheikh_Hasina - RESIGNATION_DUE_TO -> Bangladesh_Protests_Of_2022–24
Sheikh_Hasina - PREDECESSOR -> Muhammad_Yunus
Student–People'S_Uprising - OVERTHROWN -> Sheikh_Hasina
Non-Cooperation_Movement - RESULTED_IN_RESIGNATION -> Sheikh_Hasina
Muhammad_Yunus - LEADER -> Interim_Government_Of_Bangladesh
Muhammad_Yunus - INTERIM_LEADER -> Anti-Discrimination_Students_Movement
Anti-Discrimination_Students_Movement - PROPOSED -> Muhammad_Yunus
Mohammed_Shahabuddin - ADMINISTERED_THE_OATH -> Muhammad_Yunus
Anti-Discrimination_Students_Movement - CALLEDTOLEAD -> Muhammad_Yunus
Hasnat_Abdullah - REQUESTED_TO_HEAD -> Muhammad_Yunus

HISTORICAL_CONTEXT:
Sheikh Hasina's Enemies:
Student Movements
Anti-Awami League Factions
Bangladesh Nationalist Party (BNP)
Jamaat-e-Islami
Civil Society Activists
Muhammad Yunus

Sheikh Hasina's Allies:
Awami League Loyalists
Chhatra League (Student wing of the Awami League)
Groups Supporting the Liberation War's Legacy
Supporters Valuing the Awami League's Historical Role



Tweet: Cancel your Netflix subscription and donate to @realDonaldTrump's campaign for entertainment #Trump #SemST

Original Stance: Against Donald Trump

Response After Zero Shot

The tweet encourages canceling a Netflix subscription to donate to Donald Trump's campaign, using supportive hashtags. This directly promotes financial support for Trump, indicating a stance **FAVOR** of him.

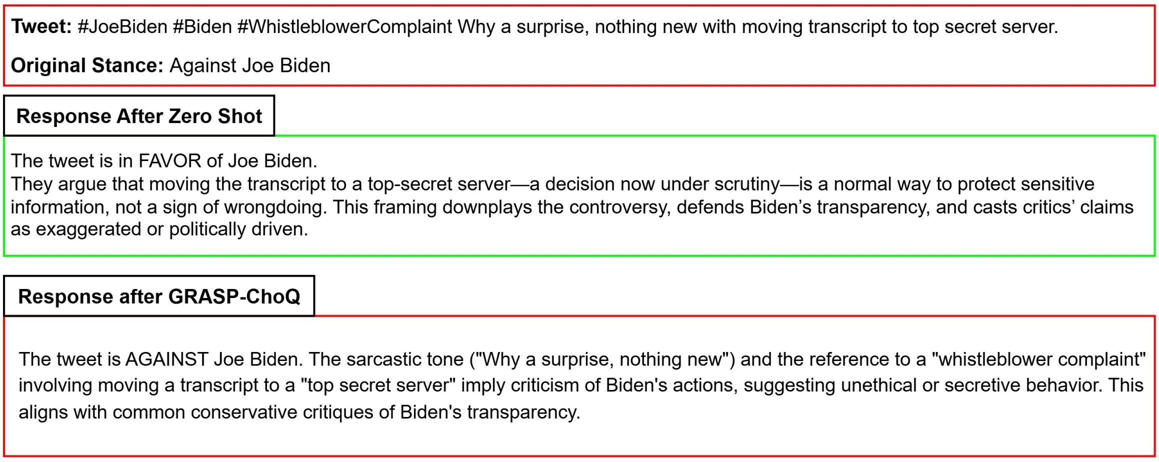
Response after GRASP-ChoQ

The tweet is **AGAINST Donald Trump**.

Key Analysis:
- **Mismatched Tone & Context:** The call to "donate to @realDonaldTrump's campaign for entertainment" uses sarcasm. Donating to a political campaign is not typically framed as "entertainment," implying that supporting Trump is absurd or laughable.
- **Trivialization:** Equating political support with "entertainment" undermines the seriousness of Trump’s campaign, suggesting it is a spectacle rather than a legitimate effort.
- **Hashtag Ambiguity:** While #Trump is neutral, the sarcastic framing inverts the literal message.

The tweet mocks Trump by treating his campaign as a joke, signaling opposition.

Figure 8: Comparison of stance classification outputs for a sarcastic tweet originally labeled as "Against Donald Trump." The Zero-Shot response misinterprets the literal content, incorrectly inferring a stance in favor of Trump. In contrast, the GRASP-ChoQ model accurately identifies sarcasm and contextual cues, correctly classifying the stance as against Trump. Key interpretive factors include mismatched tone, trivialization, and hashtag ambiguity.



Tweet: #JoeBiden #Biden #WhistleblowerComplaint Why a surprise, nothing new with moving transcript to top secret server.

Original Stance: Against Joe Biden

Response After Zero Shot

The tweet is in FAVOR of Joe Biden.
They argue that moving the transcript to a top-secret server—a decision now under scrutiny—is a normal way to protect sensitive information, not a sign of wrongdoing. This framing downplays the controversy, defends Biden’s transparency, and casts critics’ claims as exaggerated or politically driven.

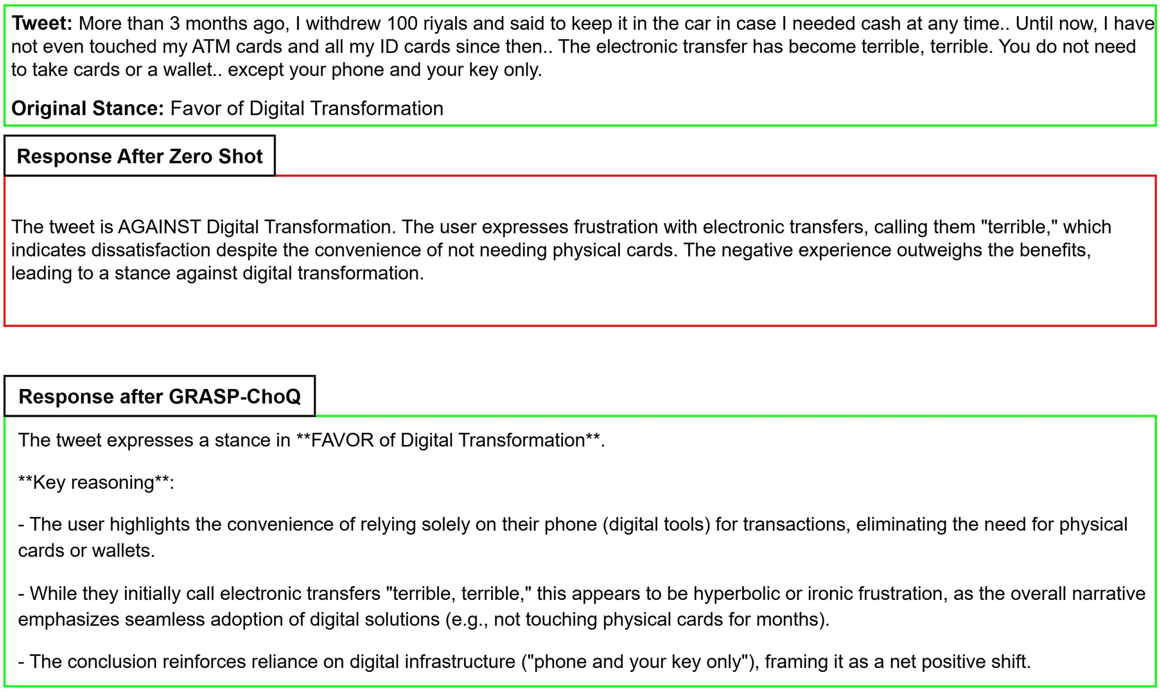
Response after GRASP-ChoQ

The tweet is AGAINST Joe Biden. The sarcastic tone ("Why a surprise, nothing new") and the reference to a "whistleblower complaint" involving moving a transcript to a "top secret server" imply criticism of Biden's actions, suggesting unethical or secretive behavior. This aligns with common conservative critiques of Biden's transparency.

Figure 9: Comparison of stance classification outputs for a sarcastic tweet originally labeled as "Against Joe Biden". The tweet is from SemEval 2016 dataset. The Zero-Shot response fails to detect sarcasm, incorrectly interpreting the tweet as supportive of Biden. In contrast, the GRASP-ChoQ model correctly infers opposition by identifying the sarcastic tone and contextual cues such as references to secrecy and a whistleblower complaint—framing consistent with common criticisms of Biden’s transparency.

C Examples of LLM Outputs

These examples highlight the significant challenges Large Language Models (LLMs) face in zero-shot stance classification, particularly when dealing with nuanced or non-literal language such as sarcasm or hyperbole. As illustrated in Figure 8 and Figure 9, the zero-shot LLM frequently misinterprets sarcastic tweets, inferring a stance directly opposite to the author’s true intent by focusing on the literal meaning while failing to detect the ironic tone and contextual cues. Similarly, Figure 10 demonstrates how a zero-shot approach can be misled by repeated negative phrasing, overlooking the broader positive sentiment and contextual nuances that indicate a favorable stance despite momentary frustration. In stark contrast, our proposed GRASP-ChoQ method consistently demonstrates a more robust understanding of



Tweet: More than 3 months ago, I withdrew 100 riyals and said to keep it in the car in case I needed cash at any time.. Until now, I have not even touched my ATM cards and all my ID cards since then.. The electronic transfer has become terrible, terrible. You do not need to take cards or a wallet.. except your phone and your key only.

Original Stance: Favor of Digital Transformation

Response After Zero Shot

The tweet is AGAINST Digital Transformation. The user expresses frustration with electronic transfers, calling them "terrible," which indicates dissatisfaction despite the convenience of not needing physical cards. The negative experience outweighs the benefits, leading to a stance against digital transformation.

Response after GRASP-ChoQ

The tweet expresses a stance in **FAVOR of Digital Transformation**.

Key reasoning:

- The user highlights the convenience of relying solely on their phone (digital tools) for transactions, eliminating the need for physical cards or wallets.

- While they initially call electronic transfers "terrible, terrible," this appears to be hyperbolic or ironic frustration, as the overall narrative emphasizes seamless adoption of digital solutions (e.g., not touching physical cards for months).

- The conclusion reinforces reliance on digital infrastructure ("phone and your key only"), framing it as a net positive shift.

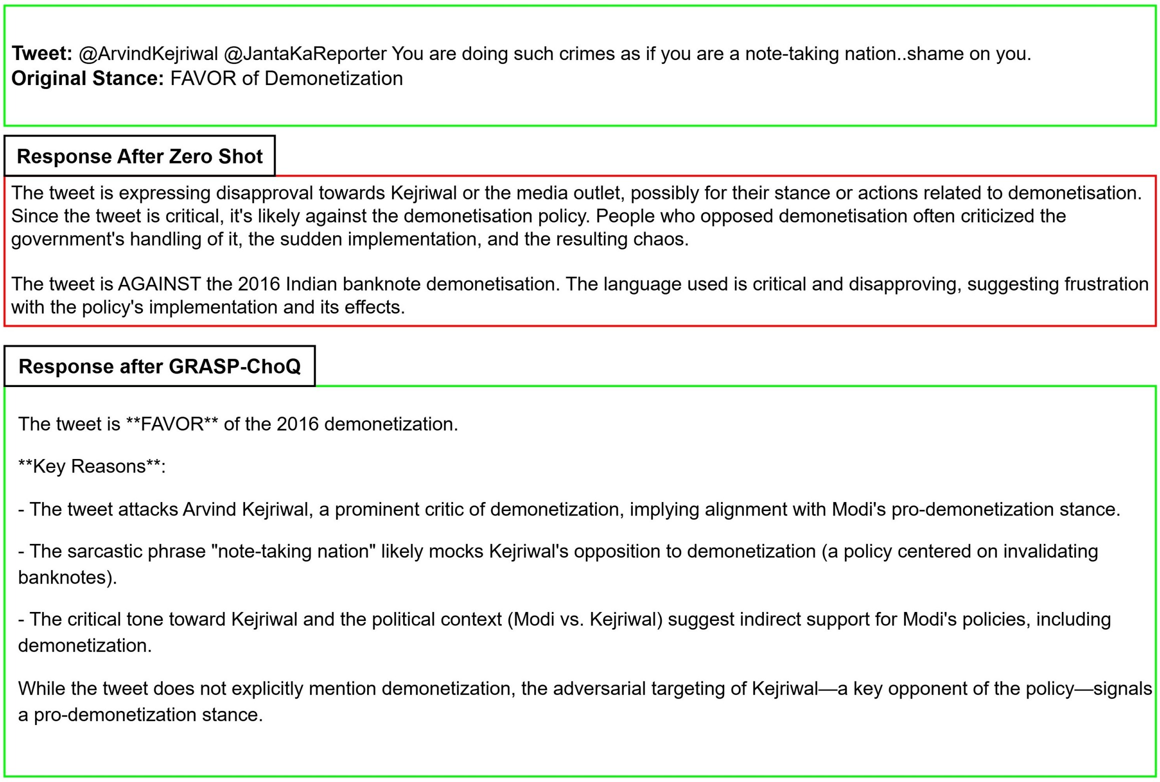
Figure 10: Comparison of stance classification for a tweet originally labeled as "In Favor of Digital Transformation". The tweet is from P-stance dataset. The Zero-Shot model misclassifies the stance as negative, focusing on the repeated phrase “terrible, terrible” and overlooking the tweet’s broader context. In contrast, GRASP-ChoQ correctly interprets the stance as favorable by recognizing the user’s emphasis on the convenience of digital transactions and framing the frustration as possibly hyperbolic or ironic. This nuanced understanding reflects support for digital infrastructure despite momentary dissatisfaction.

these complexities. By leveraging its analytical framework, GRASP-ChoQ accurately identifies sarcasm, interprets contextual references, and discerns underlying tones, leading to correct stance classifications even in challenging cases where the zero-shot LLM fails. Further examples in Figure 11, Figure 12, and Figure 13 show how GRASP-ChoQ successfully revises the zero-shot interpretation, aligning with the true stance by capturing subtleties missed by the baseline model. These comparisons underscore the limitations of zero-shot LLMs for nuanced stance detection and emphasize the superior performance achieved by incorporating the structured analysis provided by our GRASP-ChoQ approach.

D Libraries Used

In this research, we employed the `TokenTextSplitter` from the LangChain library to preprocess and split large raw text documents into smaller, manageable chunks for efficient processing by language models. We configured the `TokenTextSplitter` with a `chunk_size` of 256 tokens and a `chunk_overlap` of 50 tokens. By overlaying 50 tokens, our arrangement guaranteed contextual continuity between subsequent chunks and let us produce small enough chunks for effective processing. The raw documents were then subjected to the `split_documents` technique to produce a list of tokenized sections. The result was examined to confirm the proper amount of pieces and their semantic coherence, therefore verifying that the approach worked for downstream language model tasks.

In our research paper, we utilized the **LangChain** library, specifically the experimental module for graph transformations, to streamline the extraction and structuring of entity relationships from unstructured textual data. By employing the `LLMGraphTransformer` class, we efficiently converted documents into graph-based representations, enabling a more detailed analysis of interconnected entities. Additionally, we leveraged the **LangChain OpenAI integration** to utilize advanced language models like GPT-4, ensuring the accurate identification of nodes and relationships. This approach facilitated the



Tweet: @ArvindKejriwal @JantaKaReporter You are doing such crimes as if you are a note-taking nation..shame on you.
Original Stance: FAVOR of Demonetization

Response After Zero Shot

The tweet is expressing disapproval towards Kejriwal or the media outlet, possibly for their stance or actions related to demonetisation. Since the tweet is critical, it's likely against the demonetisation policy. People who opposed demonetisation often criticized the government's handling of it, the sudden implementation, and the resulting chaos.

The tweet is AGAINST the 2016 Indian banknote demonetisation. The language used is critical and disapproving, suggesting frustration with the policy's implementation and its effects.

Response after GRASP-ChoQ

The tweet is **FAVOR** of the 2016 demonetization.

Key Reasons:

- The tweet attacks Arvind Kejriwal, a prominent critic of demonetization, implying alignment with Modi's pro-demonetization stance.

- The sarcastic phrase "note-taking nation" likely mocks Kejriwal's opposition to demonetization (a policy centered on invalidating banknotes).

- The critical tone toward Kejriwal and the political context (Modi vs. Kejriwal) suggest indirect support for Modi's policies, including demonetization.

While the tweet does not explicitly mention demonetization, the adversarial targeting of Kejriwal—a key opponent of the policy—signals a pro-demonetization stance.

Figure 11: Comparison of tweet interpretations concerning the 2016 Indian demonetization from (Swami et al., 2018). The top section presents the original tweet and its initial stance. The middle section shows an interpretation ("Response After Zero Shot") indicating opposition to demonetization. The bottom section ("Response After GRASP-ChoQ") reveals a revised interpretation favoring demonetization after applying the GRASP-ChoQ analytical framework.

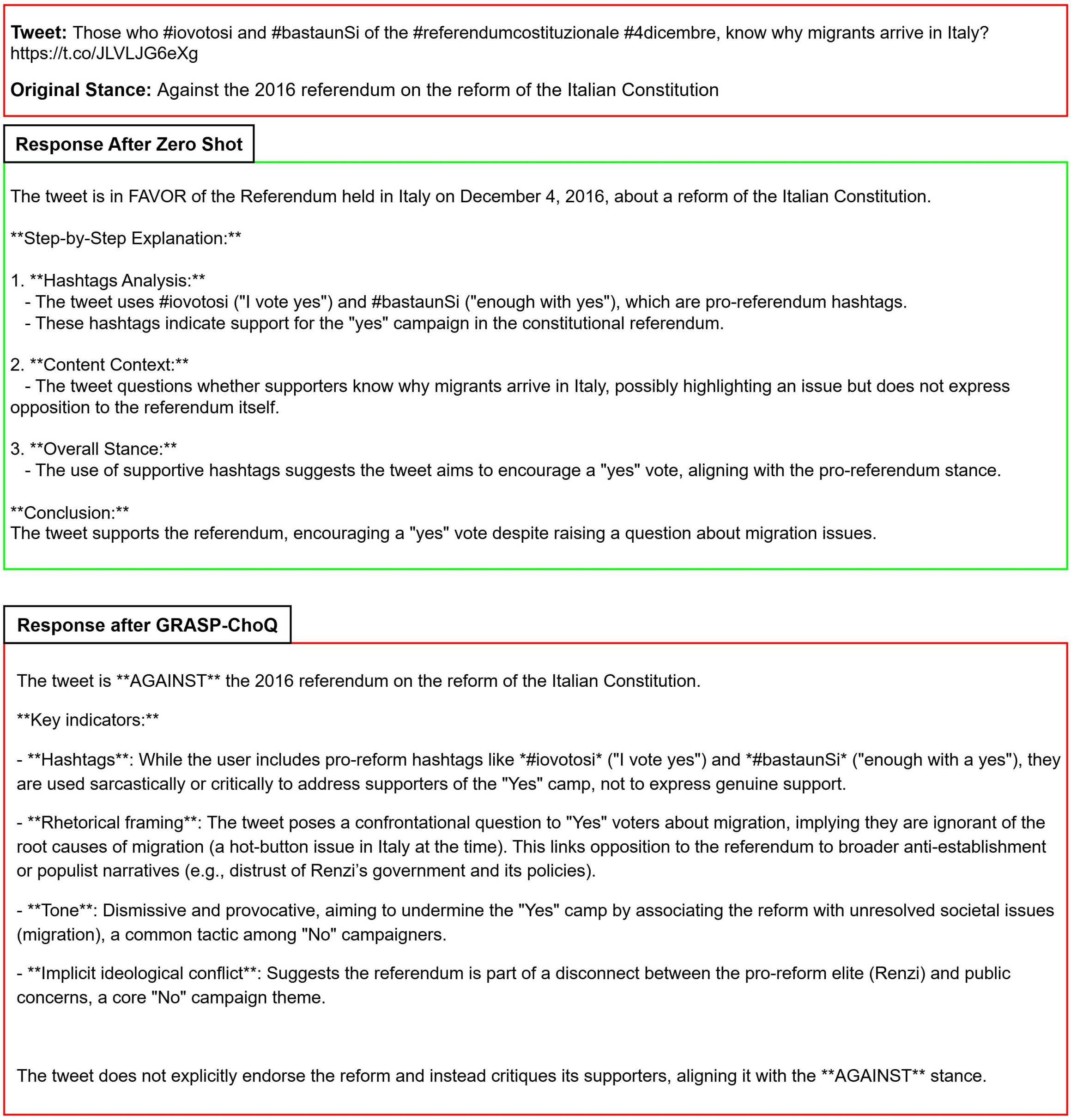
generation of knowledge graphs that supported our research objectives by providing a structured and interpretable format for complex datasets.

This study produced dense vector representations of text using the pre-trained model `sentence-transformers/all-MiniLM-L6-v2` by way of the `Hugging FaceEmbeddings` class from LangChain. These embeddings were coupled with a Neo4j graph database using the `Neo4jVector` class to enable hybrid search capabilities, integrating semantic correspondence alongside graph traversal. We extracted embeddings from network nodes labeled as "Document" and employed precomputed embeddings and text content to allow efficient and context-aware retrieval of information. This method leverages the structured relationships of graph databases to harness the power of semantic embeddings.

This approach generates Lucene-compatible full-text search queries for Neo4j by utilizing the `remove_lucene_chars` method from the `langchain_community.vectorstores.neo4j_vector` library to enable approximate matching with fuzzy search. The initial step sanitizes the input text by removing special characters that could interfere with Lucene syntax. Next, the sanitized text is split into individual words, and each word is appended with the fuzzy search operator (∼2) to allow a Levenshtein distance of up to 2, accommodating minor spelling errors or variations. These words are then combined using the `AND` operator to ensure all terms appear in the search results with fuzzy matching. This robust query-generation method is particularly valuable for Neo4j graph databases with Lucene-based indexing in applications requiring error-tolerant and flexible search capabilities.

E Neo4j Workflow

We created a function `structured_retriever` to process an input tweet and return a string containing structured data extracted from a graph



Tweet: Those who #iovotosi and #bastaunSi of the #referendumcostituzionale #4dicembre, know why migrants arrive in Italy? https://t.co/JLVLJG6eXg

Original Stance: Against the 2016 referendum on the reform of the Italian Constitution

Response After Zero Shot

The tweet is in FAVOR of the Referendum held in Italy on December 4, 2016, about a reform of the Italian Constitution.

Step-by-Step Explanation:

1. **Hashtags Analysis:**
   - The tweet uses #iovotosi ("I vote yes") and #bastaunSi ("enough with yes"), which are pro-referendum hashtags.
   - These hashtags indicate support for the "yes" campaign in the constitutional referendum.

2. **Content Context:**
   - The tweet questions whether supporters know why migrants arrive in Italy, possibly highlighting an issue but does not express opposition to the referendum itself.

3. **Overall Stance:**
   - The use of supportive hashtags suggests the tweet aims to encourage a "yes" vote, aligning with the pro-referendum stance.

Conclusion:
The tweet supports the referendum, encouraging a "yes" vote despite raising a question about migration issues.

Response after GRASP-ChoQ

The tweet is **AGAINST** the 2016 referendum on the reform of the Italian Constitution.

Key indicators:

- **Hashtags**: While the user includes pro-reform hashtags like *#iovotosi* ("I vote yes") and *#bastaunSi* ("enough with a yes"), they are used sarcastically or critically to address supporters of the "Yes" camp, not to express genuine support.

- **Rhetorical framing**: The tweet poses a confrontational question to "Yes" voters about migration, implying they are ignorant of the root causes of migration (a hot-button issue in Italy at the time). This links opposition to the referendum to broader anti-establishment or populist narratives (e.g., distrust of Renzi's government and its policies).

- **Tone**: Dismissive and provocative, aiming to undermine the "Yes" camp by associating the reform with unresolved societal issues (migration), a common tactic among "No" campaigners.

- **Implicit ideological conflict**: Suggests the referendum is part of a disconnect between the pro-reform elite (Renzi) and public concerns, a core "No" campaign theme.

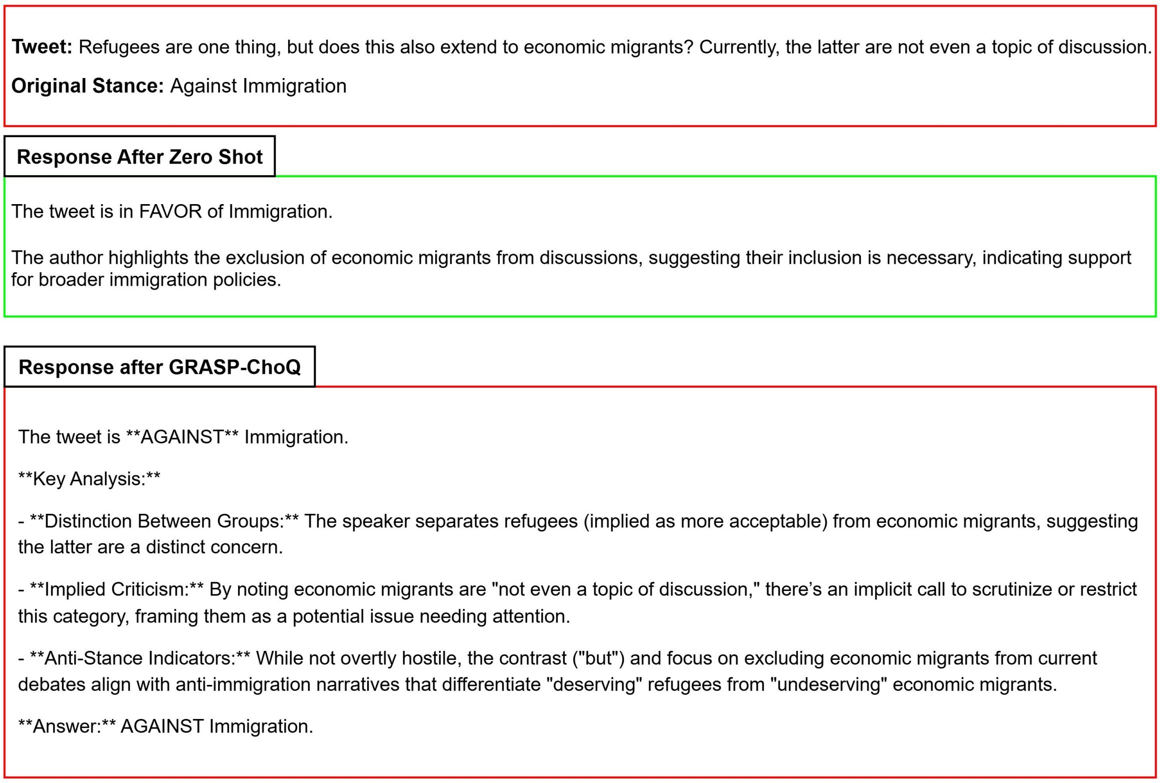
The tweet does not explicitly endorse the reform and instead critiques its supporters, aligning it with the **AGAINST** stance.

Figure 12: Comparison of tweet interpretations regarding the 2016 Italian constitutional referendum from (Lai et al., 2018). The top section displays the original tweet and its initial stance. The middle section presents an interpretation ("Response After Zero Shot") indicating support for the referendum. The bottom section ("Response After GRASP-ChoQ") reveals a revised interpretation opposing the referendum after applying the GRASP-ChoQ analytical framework.

database. Next, the function invokes an entity extraction process via `entity_chain.invoke` to identify key entities within the question. For each extracted entity, it generates a full-text search query using `generate_full_text_query` and executes a corresponding Cypher query on the graph database. This query retrieves nodes that match the entity from a full-text index and then traverses the graph to find related nodes via relationships, deliberately excluding those of type `MENTIONS`. The results from the query—formatted to include node identifiers and relationship types—are concatenated together into a single string, which is returned as the final output.

F Supplementary Datasets History

Extensive data gathered during the student-led protest movement in Bangladesh, spanning **July 15 to August 5, 2024**, marks a turning point in the political history of the country. This



Tweet: Refugees are one thing, but does this also extend to economic migrants? Currently, the latter are not even a topic of discussion.

Original Stance: Against Immigration

Response After Zero Shot

The tweet is in FAVOR of Immigration.

The author highlights the exclusion of economic migrants from discussions, suggesting their inclusion is necessary, indicating support for broader immigration policies.

Response after GRASP-ChoQ

The tweet is **AGAINST** Immigration.

Key Analysis:

- **Distinction Between Groups:** The speaker separates refugees (implied as more acceptable) from economic migrants, suggesting the latter are a distinct concern.

- **Implied Criticism:** By noting economic migrants are "not even a topic of discussion," there's an implicit call to scrutinize or restrict this category, framing them as a potential issue needing attention.

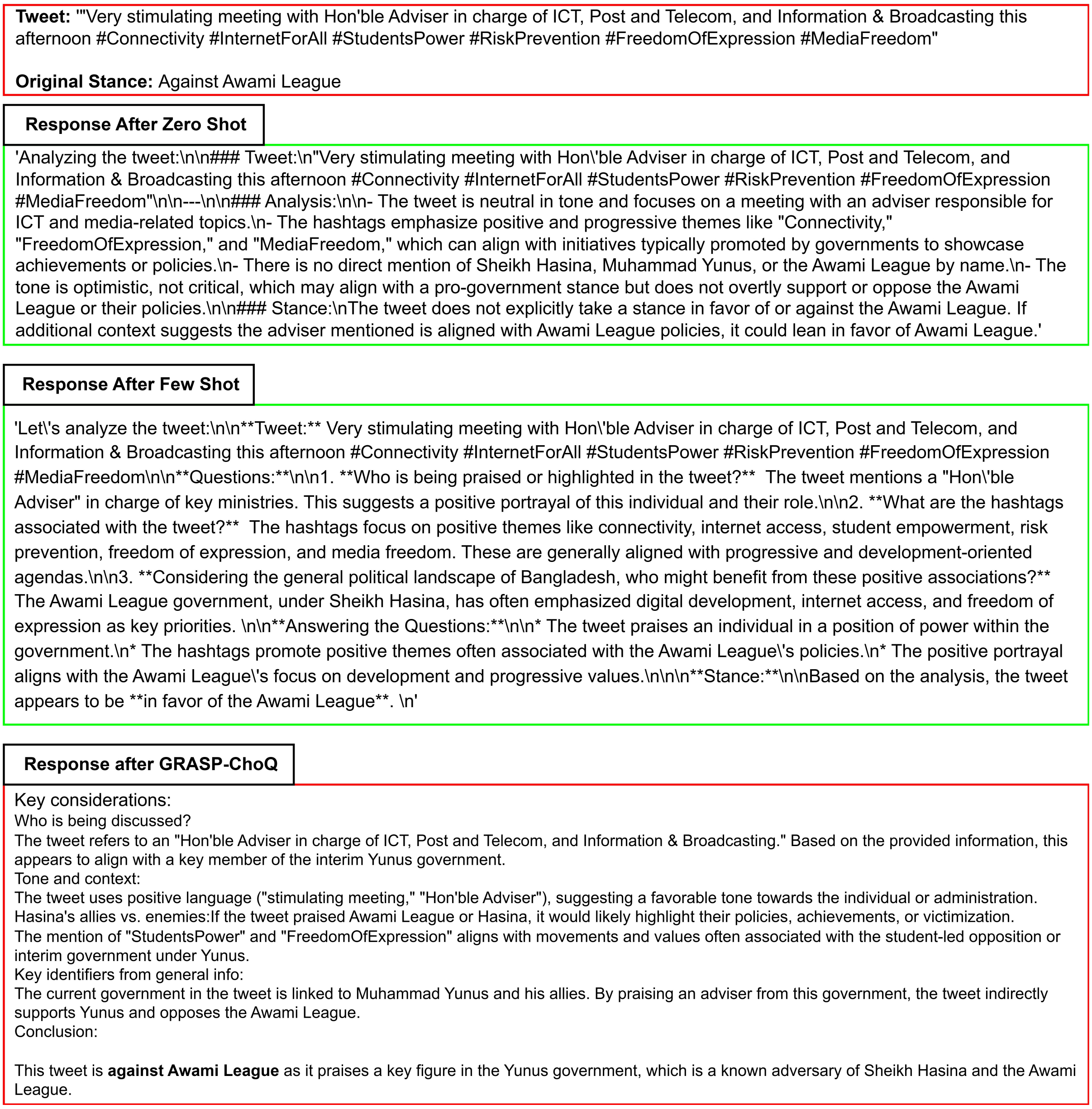
- **Anti-Stance Indicators:** While not overtly hostile, the contrast ("but") and focus on excluding economic migrants from current debates align with anti-immigration narratives that differentiate "deserving" refugees from "undeserving" economic migrants.

Answer: AGAINST Immigration.

Figure 13: Comparison of tweet interpretations regarding immigration from (Lüüsi et al., 2024). The top section presents the original tweet and its initial stance. The middle section shows an interpretation ("Response After Zero Shot") indicating support for immigration. The bottom section ("Response After GRASP-ChoQ") reveals a revised interpretation opposing immigration after applying the GRASP-ChoQ analytical framework.

research effort extracts the data. Inspired by university students opposing the reintroduction of a 30% government job quota for the children of independence war veterans a measure revoked by the Supreme Court in late June 2024 the movement rapidly acquired momentum. Originally subdued, the demonstrations—mostly involving students of Dhaka University—became more intense when members of the ruling Awami League's student branch, the Bangladesh Chatra League (BCL), assaulted protestors. With Abu Sayed, a student shot by police, rising as a symbol of opposition, police actions involving tear gas and live bullets led to over 200 deaths and thousands of injuries. From particular quota changes to more general demands for government responsibility and human rights, the movement's emphasis changed and resulted in the Student-People's Uprising, which inspired popular support across social lines. A large-scale protest in Dhaka on August 5, 2024, drove Prime Minister Sheikh Hasina to resign under government crackdowns and worldwide criticism of violations of human rights. Honoring the resiliency and tenacity of its youth-led movement, Bangladesh was named The **Economist's "Country of the Year** for 2024, acknowledged for its great influence on democracy (The Economist, 2024).

.



Tweet: "'Very stimulating meeting with Hon'ble Adviser in charge of ICT, Post and Telecom, and Information & Broadcasting this afternoon #Connectivity #InternetForAll #StudentsPower #RiskPrevention #FreedomOfExpression #MediaFreedom"

Original Stance: Against Awami League

Response After Zero Shot

'Analyzing the tweet:\n\n### Tweet:\n"Very stimulating meeting with Hon\'ble Adviser in charge of ICT, Post and Telecom, and Information & Broadcasting this afternoon #Connectivity #InternetForAll #StudentsPower #RiskPrevention #FreedomOfExpression #MediaFreedom"\n\n---\n\n### Analysis:\n\n- The tweet is neutral in tone and focuses on a meeting with an adviser responsible for ICT and media-related topics.\n- The hashtags emphasize positive and progressive themes like "Connectivity," "FreedomOfExpression," and "MediaFreedom," which can align with initiatives typically promoted by governments to showcase achievements or policies.\n- There is no direct mention of Sheikh Hasina, Muhammad Yunus, or the Awami League by name.\n- The tone is optimistic, not critical, which may align with a pro-government stance but does not overtly support or oppose the Awami League or their policies.\n\n### Stance:\nThe tweet does not explicitly take a stance in favor of or against the Awami League. If additional context suggests the adviser mentioned is aligned with Awami League policies, it could lean in favor of Awami League.'

Response After Few Shot

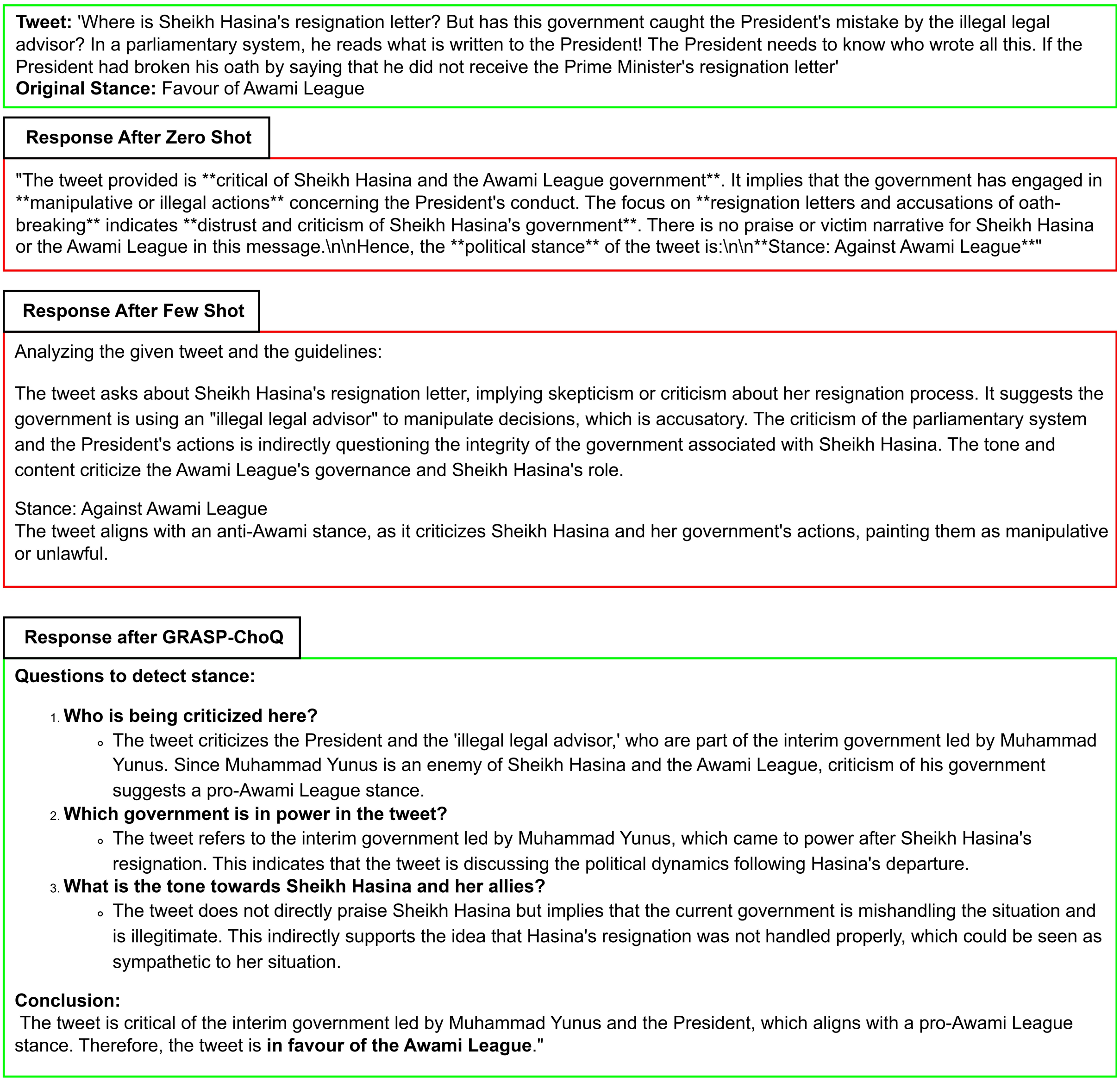
'Let\'s analyze the tweet:\n\n**Tweet:** Very stimulating meeting with Hon\'ble Adviser in charge of ICT, Post and Telecom, and Information & Broadcasting this afternoon #Connectivity #InternetForAll #StudentsPower #RiskPrevention #FreedomOfExpression #MediaFreedom\n\n**Questions:**\n\n1. **Who is being praised or highlighted in the tweet?** The tweet mentions a "Hon\'ble Adviser" in charge of key ministries. This suggests a positive portrayal of this individual and their role.\n\n2. **What are the hashtags associated with the tweet?** The hashtags focus on positive themes like connectivity, internet access, student empowerment, risk prevention, freedom of expression, and media freedom. These are generally aligned with progressive and development-oriented agendas.\n\n3. **Considering the general political landscape of Bangladesh, who might benefit from these positive associations?** The Awami League government, under Sheikh Hasina, has often emphasized digital development, internet access, and freedom of expression as key priorities. \n\n**Answering the Questions:**\n\n* The tweet praises an individual in a position of power within the government.\n* The hashtags promote positive themes often associated with the Awami League\'s policies.\n* The positive portrayal aligns with the Awami League\'s focus on development and progressive values.\n\n\n**Stance:**\n\nBased on the analysis, the tweet appears to be **in favor of the Awami League**. \n'

Response after GRASP-ChoQ

Key considerations:
Who is being discussed?
The tweet refers to an "Hon'ble Adviser in charge of ICT, Post and Telecom, and Information & Broadcasting." Based on the provided information, this appears to align with a key member of the interim Yunus government.
Tone and context:
The tweet uses positive language ("stimulating meeting," "Hon'ble Adviser"), suggesting a favorable tone towards the individual or administration.
Hasina's allies vs. enemies:If the tweet praised Awami League or Hasina, it would likely highlight their policies, achievements, or victimization.
The mention of "StudentsPower" and "FreedomOfExpression" aligns with movements and values often associated with the student-led opposition or interim government under Yunus.
Key identifiers from general info:
The current government in the tweet is linked to Muhammad Yunus and his allies. By praising an adviser from this government, the tweet indirectly supports Yunus and opposes the Awami League.
Conclusion:

This tweet is **against Awami League** as it praises a key figure in the Yunus government, which is a known adversary of Sheikh Hasina and the Awami League.

Figure 14: Using zero-shot, few-shot, and GRASP-ChoQ methods, this chart displays stance categorization findings. Zero-shot marks the tweet as neutral, while few-shot indicates a pro-Awami League stance. GRASP-ChoQ coincides with the original label, marking it as opposed to the Awami League.



Tweet: 'Where is Sheikh Hasina's resignation letter? But has this government caught the President's mistake by the illegal legal advisor? In a parliamentary system, he reads what is written to the President! The President needs to know who wrote all this. If the President had broken his oath by saying that he did not receive the Prime Minister's resignation letter'
Original Stance: Favour of Awami League

Response After Zero Shot

"The tweet provided is **critical of Sheikh Hasina and the Awami League government**. It implies that the government has engaged in **manipulative or illegal actions** concerning the President's conduct. The focus on **resignation letters and accusations of oath-breaking** indicates **distrust and criticism of Sheikh Hasina's government**. There is no praise or victim narrative for Sheikh Hasina or the Awami League in this message.\n\nHence, the **political stance** of the tweet is:\n\n**Stance: Against Awami League**"

Response After Few Shot

Analyzing the given tweet and the guidelines:

The tweet asks about Sheikh Hasina's resignation letter, implying skepticism or criticism about her resignation process. It suggests the government is using an "illegal legal advisor" to manipulate decisions, which is accusatory. The criticism of the parliamentary system and the President's actions is indirectly questioning the integrity of the government associated with Sheikh Hasina. The tone and content criticize the Awami League's governance and Sheikh Hasina's role.

Stance: Against Awami League
The tweet aligns with an anti-Awami stance, as it criticizes Sheikh Hasina and her government's actions, painting them as manipulative or unlawful.

Response after GRASP-ChoQ

Questions to detect stance:

1. **Who is being criticized here?**
   - The tweet criticizes the President and the 'illegal legal advisor,' who are part of the interim government led by Muhammad Yunus. Since Muhammad Yunus is an enemy of Sheikh Hasina and the Awami League, criticism of his government suggests a pro-Awami League stance.
2. **Which government is in power in the tweet?**
   - The tweet refers to the interim government led by Muhammad Yunus, which came to power after Sheikh Hasina's resignation. This indicates that the tweet is discussing the political dynamics following Hasina's departure.
3. **What is the tone towards Sheikh Hasina and her allies?**
   - The tweet does not directly praise Sheikh Hasina but implies that the current government is mishandling the situation and is illegitimate. This indirectly supports the idea that Hasina's resignation was not handled properly, which could be seen as sympathetic to her situation.

Conclusion:
The tweet is critical of the interim government led by Muhammad Yunus and the President, which aligns with a pro-Awami League stance. Therefore, the tweet is **in favour of the Awami League**."

Figure 15: An instance of a misclassified stance in BPDisC dataset. Stance categorization results are compared here utilizing zero-shot, few-shot, and GRASP-ChoQ approaches. Although zero-shot and few-shot label the tweet as against the Awami League, GRASP-ChoQ rightly notes that it favors the Awami League, so matching with the initial label.

Reveal-Bangla: A Dataset for Cross-Lingual Multi-Step Reasoning Evaluation

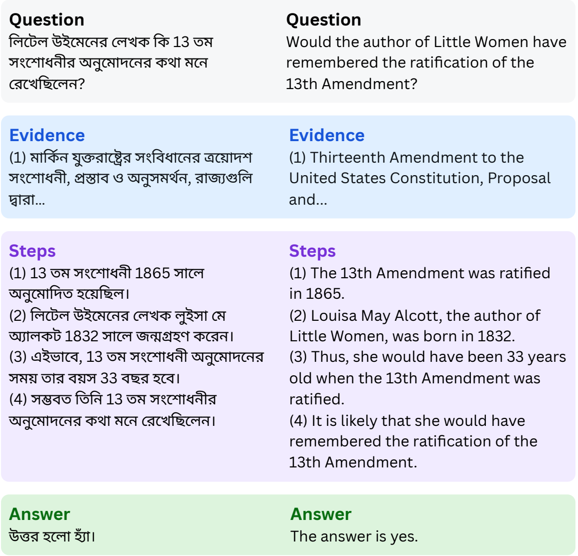
Khondoker Ittehadul Islam **Gabriele Sarti**
Center for Language and Cognition (CLCG), University of Groningen
k.i.islam@student.rug.nl g.sarti@rug.nl

Abstract

Language models have demonstrated remarkable performance on complex multi-step reasoning tasks. However, their evaluation has been predominantly confined to high-resource languages such as English. In this paper, we introduce a manually translated Bangla multi-step reasoning dataset derived from the English Reveal dataset, featuring both binary and non-binary question types. We conduct a controlled evaluation of English-centric and Bangla-centric multilingual small language models on the original dataset and our translated version to compare their ability to exploit relevant reasoning steps to produce correct answers. Our results show that, in comparable settings, reasoning context is beneficial for more challenging non-binary questions, but models struggle to employ relevant Bangla reasoning steps effectively. We conclude by exploring how reasoning steps contribute to models' predictions, highlighting different trends across models and languages. [1]

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable versatility across a wide spectrum of natural language processing tasks (Radford et al., 2019). A pivotal breakthrough in enhancing their complex reasoning capabilities has been the introduction of Chain-of-Thought (CoT) prompting (Wei et al., 2022), which encourages models to generate intermediate reasoning steps before arriving at final answers, yielding substantial performance improvements (White et al., 2024; Wang et al., 2022). Despite these advances, the evaluation of LLM reasoning capabilities remains heavily skewed toward high-resource languages, creating significant gaps in our understanding of how these models perform across linguistically diverse contexts. In this work, we focus specifically on the Bangla language, which boasts 268 million speakers and ranks as the sixth most spoken language globally[2], particularly for its computationally challenging morphological richness (Choudhury et al., 2007; Das et al., 2010). As the native language of Bangladesh and the second most prominent Indo-Aryan language after Hindi (Eberhard et al., 2021), Bangla represents a critical case study for cross-lingual reasoning evaluation. The growing technological transformation in densely populated and economically emerging regions where Bangla is spoken (Rahman, 2024) underscores the urgent need for developing faithful AI technologies that can enhance social welfare and economic opportunities. While recent work in Bangla focused on simple extractive or multiple-choice question answering Ekram et al. (2022); Shafayat et al. (2024);



Question লিটেল উইমেনের লেখক কি 13 তম সংশোধনীর অনুমোদনের কথা মনে রেখেছিলেন?	**Question** Would the author of Little Women have remembered the ratification of the 13th Amendment?
Evidence (1) মার্কিন যুক্তরাষ্ট্রের সংবিধানের ত্রয়োদশ সংশোধনী, প্রস্তাব ও অনুসমর্থন, রাজ্যগুলি দ্বারা...	**Evidence** (1) Thirteenth Amendment to the United States Constitution, Proposal and...
Steps (1) 13 তম সংশোধনী 1865 সালে অনুমোদিত হয়েছিল। (2) লিটেল উইমেনের লেখক লুইসা মে অ্যালকট 1832 সালে জন্মগ্রহণ করেন। (3) এইভাবে, 13 তম সংশোধনী অনুমোদনের সময় তার বয়স 33 বছর হবে। (4) সম্ভবত তিনি 13 তম সংশোধনীর অনুমোদনের কথা মনে রেখেছিলেন।	**Steps** (1) The 13th Amendment was ratified in 1865. (2) Louisa May Alcott, the author of Little Women, was born in 1832. (3) Thus, she would have been 33 years old when the 13th Amendment was ratified. (4) It is likely that she would have remembered the ratification of the 13th Amendment.
Answer উত্তর হলো হ্যাঁ।	**Answer** The answer is yes.

Figure 1: A Row instance of **Reveal-Bangla** containing translated Question, Evidence, Reasoning Steps and Answer from Reveal.

[1] Dataset: https://huggingface.co/datasets/khondoker/reveal-bangla, licensed CC-BY-ND 4.0

[2] https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 31–43
December 23, 2025 ©2025 Association for Computational Linguistics

Rony et al. (2024), to our knowledge, no datasets with human-validated reasoning steps are available for this language. This lack of resources hinders our ability to assess and improve the reasoning capabilities of LLMs in the Bangla language.

In this work, we address this gap by introducing **Reveal-Bangla**, a manually translated Bangla version of a subset of the English Reveal dataset, containing annotated multi-step reasoning chains with gold answers. We exploit our resource and its original English counterpart to evaluate the abilities of two small language models—both proficient in Bangla and English, but one predominantly English-centric, and the other mainly Bangla-centric—in exploiting reasoning step to produce the correct answers given a query, following recent work showing how non-English languages can harm reasoning abilities in LLMs (Qi et al., 2025).

Moreover, recent cross-lingual studies have revealed that generated reasoning chains often exhibit inconsistencies and produce misleading intermediate steps, raising questions about their explanatory reliability (Lanham et al., 2023; Paul et al., 2024). To address these concerns, post-hoc attribution techniques have emerged as valuable tools for analyzing models' internal processes by assigning importance scores to context elements such as summarization (Varun et al., 2024) and retrieved documents (Qi et al., 2024; Cohen-Wang et al., 2024), thereby revealing their contribution to final predictions. We exploit a similar methodology using the ContextCite method (Cohen-Wang et al., 2024) to examine how reasoning steps contribute to model answers in English and Bangla, highlighting different patterns of importance across the two languages.

2 Related Work

Large Language Models (LLMs) operate as probabilistic sequence predictors, estimating the likelihood of the next token given previous context (Vaswani et al., 2017; Radford et al., 2019). For practical application, explicit training on instructions was found to further improve answer quality (Sanh et al., 2022; Wang et al., 2022). Recently, eliciting reasoning from LLMs, e.g. via step-by-step Chain of Thought reasoning (CoT, Wei et al., 2022), was found to further improve the response accuracy for complex queries.

Some popular reasoning datasets in English include StrategyQA (Geva et al., 2021), featuring reasoning- and knowledge-intensive yes/no queries; Fermi (Kalyan et al., 2021), comprising estimation questions that require numerical answers and a blend of knowledge and reasoning; MuSiQue (Trivedi et al., 2022), which includes multi-hop reasoning questions with free-text entity answers, generated from Wikipedia paragraphs; and Sports Understanding (Srivastava et al., 2022), consisting of yes/no questions that demand reasoning about sports players, leagues, and maneuvers. Jacovi et al. (2024) combined the aforementioned datasets and human-annotated each LLM-generated step in terms of **attribution** relative to provided Wikipedia paragraphs and **logical coherence** in light of previous reasoning steps. The resulting dataset, dubbed Reveal, was used to prompt capable LLMs in a chain-of-thought setting and analyzed using Natural Language Inference (NLI) classifiers to evaluate model-generated responses. In our work, we manually translate a subset of Reveal into Bangla and adopt their setup to evaluate cross-lingual English-Bangla models.

Recently, Jin et al. (2024) explored small and large parameterized models, revealing a linear relationship between accuracy and the number of reasoning steps. We conduct a similar analysis, focusing particularly on small language models (SLMs) with a manageable size (1B parameters). SLMs were recently found capable of high-quality answers in RAG setups (Huang et al., 2024; Liu et al., 2024b), with relevant input information compensating for their limited reasoning abilities. In this work, we use annotated CoTs produced by larger LLMs to investigate whether SLMs can effectively leverage reasoning information in English and Bangla.

3 Development of Reveal-Bangla

3.1 Data Collection

We start by selecting a subset of the Reveal dataset.[3] Provided we want to test the ability of SLMs to obtain the correct answer given valid reasoning chains, we focus specifically on examples having all reasoning steps as either logical or fully attributable to the provided Wikipedia paragraphs. Furthermore, among the three models considered by the Reveal authors to generate answers, we decided to choose the two models with the most answers, i.e., Flan-UL2-20B (Tay et al., 2022) and

[3] https://huggingface.co/datasets/google/reveal

GPT-3 (text-davinci-003, Brown et al., 2020).[4]

We obtain a total of 104 unique questions, with 188 evidence paragraphs and 355 reasoning steps. While only 60% of all the steps are fully attributed to context, all steps are logically relevant. The dataset contains 70% yes–no *binary* questions, making it especially fitting for verifying the relevance of reasoning steps towards a simple atomic answer.[5]

Translation The English→Bangla translation of the selected subset (751 texts) was performed by a native Bangla-speaking graduate student. During the translation process, some digits and certain terms were left unchanged, for instance *76ers* (a basketball team in the NBA), *g/dL* (Grams per decilitre), /kævənd/ (pronunciation of Henry Cavendish), *Équipe d'Haïti de football* (French spelling of the Haitian National Football Team), *inter alia*. As an additional analysis, we assess the quality of automatic translations from Google Translate on the same subset, finding high-quality outputs for health and historical data, but subpar performance on the Sports Understanding subset (examples in Appendix D). Generally, automatic translations were of higher quality when performed one sentence at a time. We employ only the manually translated subset in our evaluation.

4 Evaluation

Model Selection For our evaluation on Reveal and our Bangla variant, we use Llama-3.2-1B-Instruct (or *EngLlama*) (Grattafiori et al., 2024) and BanglaLLama-3.2-1b-bangla-alpaca-orca-instruct-v0.0.1 (or *BenLlama*) (Zehady et al., 2024). *EngLlama* is a popular English-centric multilingual SLM, while *BenLlama* is a Bangla-centric model fine-tuned from *EngLlama* using Bangla-Alpaca-Orca, a collection of instruction tuning examples including the popular Alpaca and OpenOrca datasets (Taori et al., 2023; Lian et al., 2023) automatically translated into Bangla. Importantly, despite their different language focus, both models maintain answering capabilities in both English and Bangla, motivating our cross-lingual analysis.

Prompting Setup We experiment our methods on two main settings: (1) gen_ans, where the model produces an answer without any reasoning step and (2) w_cot_gen_ans, where we provide the model with the annotated reasoning steps from Reveal and Reveal-Bangla.[6] Both models were tested on the English and Bangla Reveal subsets containing the same examples, using a prompt including the query and relevant evidence paragraphs, plus the reasoning steps in the w_cot_gen_ans setting.[7] We test our models on Nvidia A100 GPU, using greedy decoding for reproducible results, and limiting output length to 256 tokens. Reasoning steps in the w_cot_gen_ans setting are appended to the assistant portion of the chat, using continue_final_message = True to let the model complete the generation by producing a final answer. We leave the remaining generation parameters unchanged.

Verifiers To verify the accuracy of the model-generated final answer against the actual final answer, we choose mDeBERTa-v3-base-xnli-multilingual-nli-2mil7 (Laurer et al., 2022) model as it is the only NLI model that supports both English and Bangla. We consider *entailment* labels as correct answers and *contradict* as otherwise. As this NLI model additionally verdicts *neutral*, authors manually verify the response to classify it as valid or not. Furthermore, as language detection tools such as langdetect (Shuyo, 2010) do not support Bangla, we manually assign *contradict* to answers generated in scripts that do not match English or Bangla in the respective settings. We provide our *hypothesis* and *premise* NLI template in the Appendix C.1. We also present additional limitations of the multi-lingual NLI model in the Appendix C.2, to foster research on cross-lingual NLI comprising Bangla.

Results Figure 2 shows the accuracy of tested models in both languages. Unsurprisingly, we find both models performed better on their respective main languages. Moreover, despite their small size, both models were generally found to effectively use the provided reasoning steps to further improve their accuracy. However, we observe that *EngLlama* obtains worse performances when given w_cot_gen_ans steps in Bangla (35.6% →

[4] We do not include Flan-PaLM-540B (Longpre et al., 2023) due to our limited evaluation resources.

[5] We present the counts and tokens distribution of steps and evidence in Figures 5 and 6 in the Appendix.

[6] A third setting prompting the model to generate its own reasoning steps, gen_cot_ans, was not included due to the poor performance of SLMs on CoT reasoning.

[7] Examples of prompt templates for each setting are available in Appendix B.

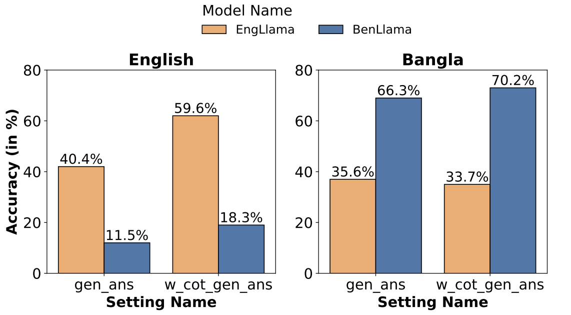


Figure 2: Accuracy of *EngLlama* and *BenLlama* for the gen_ans and w_cot_gen_ans settings on English and Bangla REVEAL subsets.

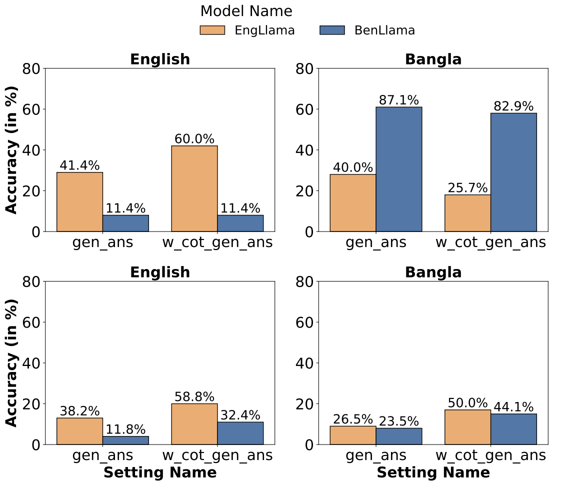


Figure 3: *EngLlama* and *BenLlama* accuracy on REVEAL Binary (top) and Non-Binary (bottom) questions.

33.7%), and find that CoT gains for the *BenLlama* model in Bangla are much milder than for the *EngLlama* model in English (+3.9% vs. +19.2%). These results confirm that, in the less-resourced Bangla setting, *additional relevant reasoning information may not be sufficient to mitigate the limited language capabilities of the tested SLMs*, especially when a Bangla-specific tuning was not performed, as was the case for *EngLlama*.

We further examine model performances across on binary and non-binary questions in the selected REVEAL subset in Figure 3. We find that the *EngLlama* model excels in non-binary questions across both languages, outperforming the *BenLlama* model in both gen_ans and w_cot_gen_ans settings, even in Bangla by a narrow margin. The stronger performance of *BenLlama* in the aggregate case is largely motivated by binary questions, in which the model obtains accuracy $> 80\%$. We also find that while CoT steps have an uneven effect on binary questions, they are consistently beneficial for non-binary ones, across

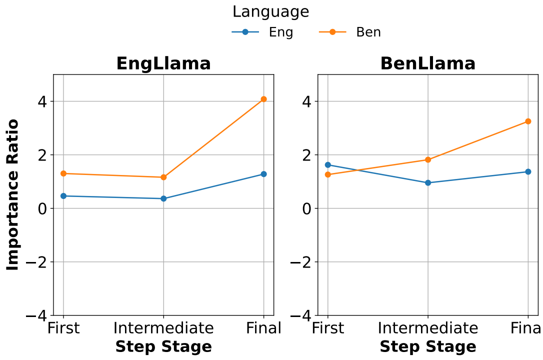


Figure 4: Importance ratio for *EngLlama* and *BenLlama* on w_cot_gen_ans reasoning steps between -4 (lowest) and $+4$ (highest).

both models and languages. This confirms previous findings on the limited effectiveness of CoT in simpler settings by Liu et al. (2024a), and suggests the benefits of CoT generalize even to less-resourced languages.

Attributing Answers to Reasoning Steps To conclude our analysis, we conduct a preliminary investigation into how CoT steps influence model answers. We employ ContextCite (Cohen-Wang et al., 2024) to attribute the final answer generated by the model to the provided reasoning steps in the w_cot_gen_ans setting using surrogate linear models, an approach similar to LIME (Ribeiro et al., 2016). Figure 4 presents an overview of our results for the two models across both languages. We observe that in most cases, later steps tend to have a larger influence on the model response. This suggests that the models place higher emphasis on answer-specific information located in later steps more than on understanding the context provided in earlier steps. This highlights the inherent limitations of these models in context comprehension, which is essential for answering complex questions. Future research could investigate whether this trend holds with larger model sizes. Additionally, we find that both models accord high importance to the Bangla language. We speculate that Bangla's morphological richness causes to assign larger values across attention layers. We leave the exploration of model interpretability in low-resource languages as an interesting direction for future work.

5 Conclusion

We presented REVEAL-Bangla, a manually translated portion of the popular English multi-step reasoning dataset REVEAL. Our cross-lingual analysis of SLMs revealed limited performance gains from CoT reasoning in the less-resourced Bangla setting compared to English, with gains primarily involving more complex non-binary questions. Further investigation into attributing reasoning steps highlighted differences in importance across models and languages. These findings underscore the need for developing language-specific approaches to enhance reasoning capabilities in low-resource languages, rather than directly transferring techniques optimized for English.

Limitations

Dataset Scale and Coverage Our study is constrained by the relatively small scale of the translated dataset, comprising only 104 unique questions from the original REVEAL dataset. This limited sample size may not fully capture the diversity of reasoning patterns and linguistic phenomena present in Bangla. Additionally, the 70% skew toward binary questions may not accurately reflect real-world reasoning scenarios, potentially overestimating model performance on more complex, open-ended reasoning tasks.

Model Selection Constraints We restricted our evaluation to small language models with 1B parameters due to computational constraints. While this choice enables insights into resource-efficient deployment scenarios, it limits our understanding of how larger, more capable models might leverage Bangla reasoning steps. The exclusion of the gen_cot_ans setting, where models generate their own reasoning chains, further restricts our analysis to scenarios with gold reasoning steps, which may not reflect realistic deployment conditions.

Translation and Annotation Quality Although we employed manual translation by a native Bangla speaker, the translation was performed by a single annotator without inter-annotator agreement measures. This approach may introduce individual biases or inconsistencies in translation choices, particularly for domain-specific terminology in sports and medical contexts. The preservation of certain English terms and pronunciations, while necessary, may also affect how models process the hybrid text.

Evaluation Methodology Limitations Our reliance on the mDeBERTa-v3-base-xnli model for answer verification introduces its own limitations, as acknowledged in our appendix. The model's tendency to produce neutral verdicts required manual intervention, potentially introducing subjective judgments. Furthermore, the absence of Bangla-specific language detection tools necessitated manual script verification, which may not scale to larger evaluations.

Cross-lingual Generalization Our findings are specific to the English-Bangla language pair and may not generalize to other low-resource languages with different linguistic properties, writing systems, or relationships to English. The choice of *BenLlama*, which was fine-tuned on automatically translated instruction data, may also introduce artifacts from machine translation that affect our conclusions about Bangla reasoning capabilities.

Attribution Analysis Scope Our investigation into reasoning step attribution using ContextCite represents only a preliminary analysis. The surrogate linear model approach may not capture complex non-linear interactions between reasoning steps, and we did not explore alternative attribution methods that might reveal different patterns of step importance across languages.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Monojit Choudhury, Vaibhav Jalan, Sudeshna Sarkar, and Anupam Basu. 2007. Evolution, optimization, and language change: The case of bengali verb inflections. In *Proceedings of ninth meeting of the ACL special interest group in computational morphology and phonology*, pages 65–74.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. 2024. Contextcite: Attributing model generation to context. *Advances in Neural Information Processing Systems*, 37:95764–95807.

Dipankar Das, Santanu Pal, Tapabrata Mondal, Tanmoy Chakraborty, and Sivaji Bandyopadhyay. 2010. Automatic extraction of complex predicates in bengali. In *Proceedings of the 2010 Workshop on Multiword Expressions: from Theory to Applications*, pages 37–45.

David Ms Eberhard, David M, Gary F. Simons, and Charles D. Fennig. 2021. Languages of the world.

Syed Mohammed Sartaj Ekram, Adham Arik Rahman, Md. Sajid Altaf, Mohammed Saidul Islam, Mehrab Mustafy Rahman, Md Mezbaur Rahman, Md Azam Hossain, and Abu Raihan Mostofa Kamal. 2022. BanglaRQA: A benchmark dataset for under-resourced Bangla language reading comprehension-based question answering with diverse question-answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2518–2532, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenyu Huang, Guancheng Zhou, Hongru Wang, Pavlos Vougiouklis, Mirella Lapata, and Jeff Z. Pan. 2024. Less is more: Making smaller language models competent subgraph retrievers for multi-hop KGQA. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15787–15803, Miami, Florida, USA. Association for Computational Linguistics.

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. 2024. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. *arXiv preprint arXiv:2402.00559*.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Ashwin Kalyan, Abhinav Kumar, Arjun Chandrasekaran, Ashish Sabharwal, and Peter Clark. 2021. How much coffee was consumed during emnlp 2019? fermi problems: A new reasoning challenge for ai. *arXiv preprint arXiv:2110.14207*.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning.

Moritz Laurer, Wouter van Atteveldt, Andreu Salleras Casas, and Kasper Welbers. 2022. Less Annotating, More Classifying – Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and BERT - NLI. *Preprint*. Publisher: Open Science Framework.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and Teknium. 2023. Openorca preview1: A llama-13b model fine-tuned on small portion of openorcav1 dataset. https://huggingface.co/Open-Orca/OpenOrca-Preview1-13B.

Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. 2024a. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. *Preprint*, arXiv:2410.21333.

Suqing Liu, Zezhu Yu, Feiran Huang, Yousef Bulbulia, Andreas Bergen, and Michael Liut. 2024b. Can small language models with retrieval-augmented generation replace large language models when learning computer science? In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2024, page 388–393, New York, NY, USA. Association for Computing Machinery.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and 1 others. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15012–15032, Miami, Florida, USA. Association for Computational Linguistics.

Jirui Qi, Shan Chen, Zidi Xiong, Raquel Fernández, Danielle S. Bitterman, and Arianna Bisazza. 2025. When models reason in your language: Controlling thinking trace language comes at the cost of accuracy.

Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. 2024. Model internals-based answer attribution for trustworthy retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6037–6053, Miami, Florida, USA. Association for Computational Linguistics.

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

Riajur Rahman. 2024. Present technology strategy of bangladesh. *BDTask Blog*.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. “why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 2016*

Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pages 97–101, San Diego, California. Association for Computational Linguistics.

Md Rashad Al Hasan Rony, Sudipto Kumar Shaha, Rakib Al Hasan, Sumon Kanti Dey, Amzad Hossain Rafi, Amzad Hossain Rafi, Ashraf Hasan Sirajee, and Jens Lehmann. 2024. Banglaquad: A bengali open-domain question answering dataset.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, and 21 others. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Sheikh Shafayat, H Hasan, Minhajur Mahim, Rifki Putri, James Thorne, and Alice Oh. 2024. BEnQA: A question answering benchmark for Bengali and English. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1158–1177, Bangkok, Thailand. Association for Computational Linguistics.

Nakatani Shuyo. 2010. Language detection library for java.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, and 1 others. 2022. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Yerram Varun, Rahul Madhavan, Sravanti Addepalli, Arun Suggala, Karthikeyan Shanmugam, and Prateek Jain. 2024. Time-reversal provides unsupervised feedback to llms. *Advances in Neural Information Processing Systems*, 37:29777–29806.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, and 1 others. 2024. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*.

Abdullah Khan Zehady, Safi Al Mamun, Naymul Islam, and Santu Karmaker. 2024. Bongllama: Llama for bangla language. *arXiv preprint arXiv:2410.21200*.

Appendix

A Dataset

A.1 Sample

Question [E]	Can a Bengal cat survive eating only pancakes?
Question [B]	বেঙ্গল বিড়াল কি শুধু প্যানকেক খেয়ে বেঁচে থাকতে পারে?
Evidence [E]	1. Carnivore, Obligate carnivores: Obligate carnivores are diverse. The amphibian axolotl consumes mainly worms and larvae in its environment, but if necessary will consume algae. All felids, including the domestic cat, require a diet of primarily animal flesh and organs. Specifically, cats have high protein requirements and their metabolisms appear unable to synthesize essential nutrients such as retinol, arginine, taurine, and arachidonic acid; thus, in nature, they must consume flesh to supply these nutrients. 2. Pancake: A pancake (or hotcake, griddlecake, or flapjack) is a flat cake, often thin and round, prepared from a starch-based batter that may contain eggs, milk and butter and cooked on a hot surface such as a griddle or frying pan, often frying with oil or butter. Archaeological evidence suggests that pancakes were probably the earliest and most widespread cereal food eaten in prehistoric societies.
Evidence [B]	1. মাংসাশী, বাধ্য মাংসাশী: বাধ্য মাংসাশী বৈচিত্র্যময়। উভচর অ্যাক্সোলটল তার পরিবেশে প্রধানত কৃমি এবং লার্ভা খায়, তবে প্রয়োজনে শেওলা গ্রাস করবে। গৃহপালিত বিড়াল সহ সমস্ত ক্ষেত্রের জন্য প্রাথমিকভাবে পশুর মাংস এবং অঙ্গগুলির একটি খাদ্য প্রয়োজন। বিশেষত, বিড়ালদের উচ্চ প্রোটিনের প্রয়োজনীয়তা থাকে এবং তাদের বিপাক রেটিনল, আরজিনাইন, টাউরিন এবং অ্যারাকিডোনিক অ্যাসিডের মতো প্রয়োজনীয় পুষ্টি সংশ্লেষণ করতে অক্ষম বলে মনে হয়; এইভাবে, প্রকৃতিতে, এই পুষ্টি সরবরাহ করার জন্য তাদের অবশ্যই মাংস গ্রহণ করতে হবে। 2. প্যানকেক: একটি প্যানকেক (বা হটকেক, গ্রিডল কেক বা ফ্ল্যাপজ্যাক) একটি ফ্ল্যাট কেক, প্রায়শই পাতলা এবং গোলাকার হয়, যা একটি স্টার্চ-ভিত্তিক ব্যাটার থেকে তৈরি করা হয় যাতে ডিম, দুধ এবং মাখন থাকতে পারে এবং একটি গরম পৃষ্ঠে রান্না করা হয় যেমন একটি ভাজা বা ফ্রাইং প্যান, প্রায়শই তেল বা মাখন দিয়ে ভাজা হয়। প্রত্নতাত্ত্বিক প্রমাণগুলি থেকে জানা যায় যে প্যানকেকগুলি সম্ভবত প্রাগৈতিহাসিক সমাজে খাওয়া সবচেয়ে প্রাচীন এবং সর্বাধিক বিস্তৃত খাদ্যশস্য ছিল।
Steps [E]	1. Cats are obligate carnivores, meaning they need to eat meat to survive. 2. Pancakes are not a source of meat. 3. Thus, a Bengal cat cannot survive eating only pancakes.
Steps [B]	1. বিড়াল বাধ্যতামূলক মাংসাশী, যার অর্থ তাদের বেঁচে থাকার জন্য মাংস খেতে হবে। 2. প্যানকেক মাংসের উৎস নয়। 3. সুতরাং, একটি বেঙ্গল বিড়াল শুধুমাত্র প্যানকেক খেয়ে বাঁচতে পারে না।
Answer [E]	The answer is no.
Answer [B]	উত্তর হলো না।

Table 1: Samples from our dataset comprising of question, evidence, steps, and answer where **[E]** and **[B]** following them represents corresponding English and Bangla versions respectively.

A.2 Step Count and Token Distribution

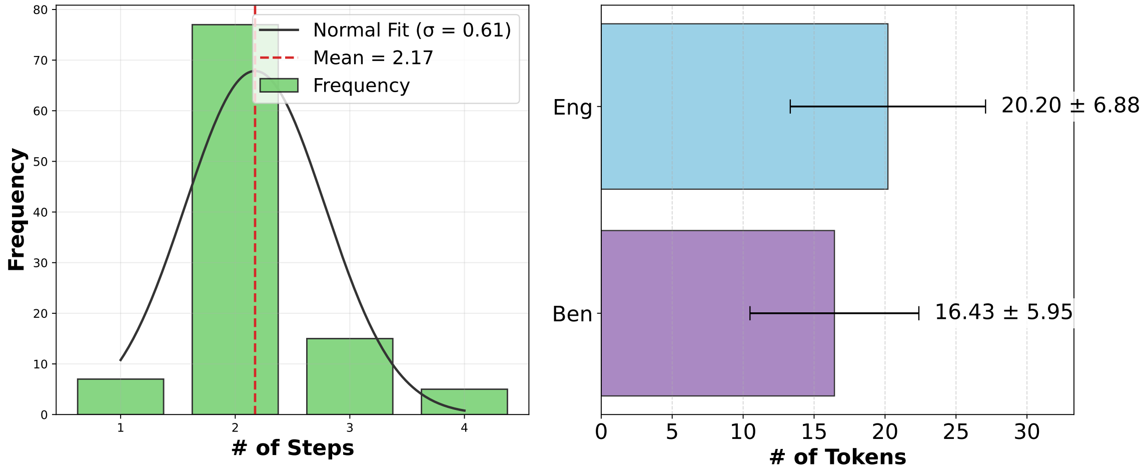


Figure 5: Distribution of Step Count and Token Distribution of Steps. Furthermore, interestingly, number of words required to describe a step in Bangla is less than of English.

A.3 Evidence Count and Token Distribution

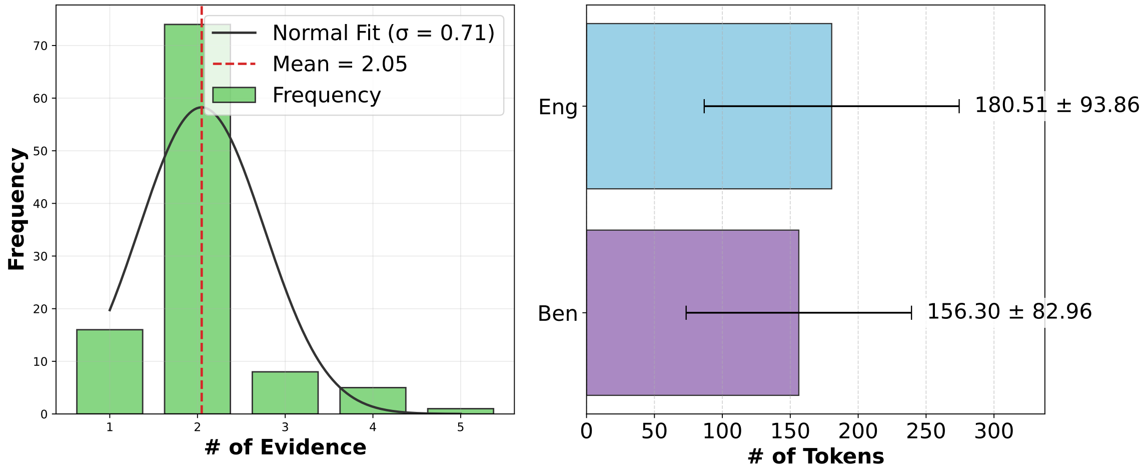


Figure 6: Distribution of Evidence Count and Token Distribution of Steps. On average, there were three evidences associated alongside the questions.

B Example of Chat Prompt Templates

B.1 Setting: gen_ans

En

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant. Your goal is to respond to user queries using the provided evidence paragraphs. The final line must contain the word 'Answer:' followed by the answer to the user query. The response should contain ONLY the final response. If the question requires a yes/no answer, answer using only "yes" or "no". Do NOT provide any additional explanation or comments.<|eot_id|><|start_header_id|>user<|end_header_id|>

# Evidence

1. Toilet paper, Description, Materials: Toilet paper is usually manufactured from pulpwood trees, but is also sometimes made from sugar cane byproducts or bamboo.

2. Logging: Logging is the process of cutting, processing, and moving trees to a location for transport. It may include skidding, on-site processing, and loading of trees or logs onto trucks or skeleton cars.

# Question:

Would it be hard to get toilet paper if there were no loggers?<|eot_id|>

<|start_header_id|>assistant<|end_header_id|>

Answer:
```

Bn

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

আপনি একজন উপকারী সহকারী। আপনার উদ্দেশ্য হল প্রদত্ত প্রমাণ অনুচ্ছেদ ব্যবহার করে ব্যবহারকারীর প্রশ্নের উত্তর দেওয়া। চূড়ান্ত লাইনে অবশ্যই 'উত্তর:' শব্দটি থাকবে এবং তারপরে ব্যবহারকারীর প্রশ্নের উত্তর থাকবে। প্রতিক্রিয়াটিতে শুধুমাত্র চূড়ান্ত উত্তর থাকবে। প্রশ্নটির যদি হ্যাঁ/না উত্তরের প্রয়োজন হয়, তাহলে শুধুমাত্র "হ্যাঁ" বা "না" ব্যবহার করে উত্তর দিন। কোন অতিরিক্ত ব্যাখ্যা বা মন্তব্য প্রদান করবেন না।<|eot_id|><|start_header_id|>user<|end_header_id|>

# প্রমাণ

1. টয়লেট পেপার, বর্ণনা, উপকরণ: টয়লেট পেপার সাধারণত পাল্পউড গাছ থেকে তৈরি করা হয়, তবে কখনও কখনও আখের উপজাত বা বাঁশ থেকেও তৈরি করা হয়।

2. লগিং: লগিং হল পরিবহনের জন্য গাছ কাটা, প্রক্রিয়াকরণ এবং স্থানান্তর করার প্রক্রিয়া। এতে স্কিডিং, অন-সাইট প্রক্রিয়াকরণ এবং ট্রাক বা এস্কেলেটন গাড়িতে গাছ বা লগ লোড করা অন্তর্ভুক্ত থাকতে পারে।

# প্রশ্ন:

কাঠুরি না থাকলে কি টয়লেট পেপার পাওয়া কঠিন হবে?<|eot_id|>

<|start_header_id|>assistant<|end_header_id|>

উত্তর:
```

Table 2: An example of a chat prompt template from gen_ans setting. **En** is of the corresponding English language and **Bn** is of the Bangla language.

B.2 Setting: w_cot_gen_ans

En

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant. Your goal is to respond to user queries using the provided evidence paragraphs. The final line must contain the word 'Answer:' followed by the answer to the user query. The response should contain ONLY the final response. If the question requires a yes/no answer, answer using only "yes" or "no". Do NOT provide any additional explanation or comments.<|eot_id|><|start_header_id|>user<|end_header_id|>
# Evidence
1. Toilet paper, Description, Materials: Toilet paper is usually manufactured from pulpwood trees, but is also sometimes made from sugar cane byproducts or bamboo.

2. Logging: Logging is the process of cutting, processing, and moving trees to a location for transport. It may include skidding, on-site processing, and loading of trees or logs onto trucks or skeleton cars.

# Question:
Would it be hard to get toilet paper if there were no loggers?<|eot_id|>

<|start_header_id|>assistant<|end_header_id|>

1. Toilet paper is made from trees.
2. Loggers are responsible for cutting down trees.

3. Thus, without loggers, it would be difficult to get toilet paper.

Answer:
```

Bn

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
আপনি একজন উপকারী সহকারী। আপনার উদ্দেশ্য হল প্রদত্ত প্রমাণ অনুচ্ছেদ ব্যবহার করে ব্যবহারকারীর প্রশ্নের উত্তর দেওয়া। চূড়ান্ত উত্তর দেওয়ার আগে, ধাপে ধাপে যুক্তি দিবেন, প্রতিটি যুক্তির ধাপকে একটি নতুন লাইনে সংখ্যাযুক্ত ভাবে তালিকাভুক্ত করবেন। চূড়ান্ত লাইনে অবশ্যই 'উত্তর:' শব্দটি থাকবে এবং তারপরে ব্যবহারকারীর প্রশ্নের উত্তর থাকবে। প্রতিক্রিয়াটিতে শুধুমাত্র সংখ্যাযুক্ত যুক্তির ধাপ এবং চূড়ান্ত উত্তর থাকবে। প্রশ্নটির যদি হ্যাঁ/না উত্তরের প্রয়োজন হয়, তাহলে শুধুমাত্র "হ্যাঁ" বা "না" ব্যবহার করে উত্তর দিন। কোন অতিরিক্ত ব্যাখ্যা বা মন্তব্য প্রদান করবেন না।<|eot_id|><|start_header_id|>user<|end_header_id|>
# প্রমাণ
1. টয়লেট পেপার, বর্ণনা, উপকরণ: টয়লেট পেপার সাধারণত পাল্পউড গাছ থেকে তৈরি করা হয়, তবে কখনও কখনও আখের উপজাত বা বাঁশ থেকেও তৈরি করা হয়।

2. লগিং: লগিং হল পরিবহনের জন্য গাছ কাটা, প্রক্রিয়াকরণ এবং স্থানান্তর করার প্রক্রিয়া। এতে স্কিডিং, অন-সাইট প্রক্রিয়াকরণ এবং ট্রাক বা এস্কেলেটন গাড়িতে গাছ বা লগ লোড করা অন্তর্ভুক্ত থাকতে পারে।

# প্রশ্ন:
কাঠুরি না থাকলে কি টয়লেট পেপার পাওয়া কঠিন হবে?<|eot_id|>

<|start_header_id|>assistant<|end_header_id|>

1. টয়লেট পেপার তৈরি হয় গাছ থেকে।
2. কাঠুরিরা গাছ কাটার জন্য দায়ী।

3. এইভাবে, কাঠুরি ছাড়া, টয়লেট পেপার পাওয়া কঠিন হবে।

উত্তর:
```

Table 3: An example of a chat prompt template from w_cot_gen_ans setting. **En** is of the corresponding English language and **Bn** is of the Bangla language.

C NLI

C.1 Structure Example

C.1.1 English

Hypothesis: Who does the actress that played mary poppins in the 1964 film play in princess diaries? The answer is Queen Clarisse Renaldi.
Premise: Who does the actress that played mary poppins in the 1964 film play in princess diaries? The answer is Julie Andrews.

Table 4: Example of the **Hypothesis** and **Premise** structure for English language. Here, **Hypothesis** incorporates ground answer and **Premise** incorporates model predicted answer.

C.1.2 Bangla

Hypothesis: নিউটন সর্বজনীন মহাকর্ষ ধ্রুবকের মান পরিমাপ করার জন্য পদার্থবিজ্ঞানীর কাজের ক্ষেত্রটি কী? সুতরাং উত্তর হলো পদার্থবিদ এবং রসায়নবিদ।
Premise: নিউটন সর্বজনীন মহাকর্ষ ধ্রুবকের মান পরিমাপ করার জন্য পদার্থবিজ্ঞানীর কাজের ক্ষেত্রটি কী? উত্তর হলো নিউটন সর্বজনীন মহাকর্ষ ধ্রুবকের মান পরিমাপ করার জন্য পদার্থবিজ্ঞানীর কাজের ক্ষেত্রটি হলো প্রমাণ।

Table 5: Example of the **Hypothesis** and **Premise** structure for Bangla language.

C.2 Example Cases of Limitations on Bangla

C.2.1 Entails Proper Noun Spelling Mistakes

Hypothesis	রেডক্যাপ অভিনেতা সদস্যের পত্নী কে? উত্তর হলো শিলা হ্যানকক।
Premise	রেডক্যাপ অভিনেতা সদস্যের পত্নী কে? উত্তর হলো শিলা হ্যানক
Ground Label	Contradiction
Predicted Label	Entailment

Table 6: Example of NLI model incorrectly predicting Entailment where the Premise differs with Hypothesis through a spelling mistake on proper noun শিলা হ্যানকক.

C.2.2 Labels *contradict* on model's elaborate correct answers on binary question

Hypothesis	ডরোথিয়া ওয়েন্ডিং কি পোর্শার উদ্ভব জায়গা থেকে এসেছেন? উত্তর হলো হ্যাঁ।
Premise	ডরোথিয়া ওয়েন্ডিং কি পোর্শার উদ্ভব জায়গা থেকে এসেছেন? উত্তর হলো ডরোথিয়া ওয়েন্ডিং পোর্শার উদ্ভব জায়গা থেকে এসেছেন।
Ground Label	Entailment
Predicted Label	Contradict

Table 7: Example of NLI model incorrectly judging Contradict where the Premise contained elaboration on the single word হ্যাঁ ("yes") answer.

C.2.3 Labels *entailment* where the main final answer is missing

Hypothesis	মহম্মদ আত্তার গাড়ি কী, যে কোম্পানির ড্যাটসান তৈরি করা হয়েছে, এর নজির? সুতরাং উত্তর হলো নিসান আলটিমা।
Premise	মহম্মদ আত্তার গাড়ি কী, যে কোম্পানির ড্যাটসান তৈরি করা হয়েছে, এর নজির? উত্তর হলো নিসান মোটর কোং, লিমিটেড (হেপবার্ন: নিসান জিদোশা কাবুশিকি গাইশা) হল একটি জাপানী বহুজাতিক অটোমোবাইল প্রস্তুতকারক যার সদর দপ্তর নিশি-কু, ইয়োকোহামা, জাপানে। কোম্পানিটি নিসান, ইনফিনিটি এবং ড্যাটসুন
Ground Label	Contradict
Predicted Label	Entailment

Table 8: Example of NLI model falsely predicts Entailment where the actual proper noun answer "নিসান আলটিমা" (Nissan Altima) is not present in the Premise.

D Google Translation Errors

D.1 Example on American Football Context

Source: DK Metcalf is an American football player. Hitting the wheel route is part of American football. So the answer is yes.
Manual Translation: ডি কে মেটকাফ একজন আমেরিকান ফুটবল খেলোয়াড়। হুইল রাউট করা আমেরিকান ফুটবলের অংশ। সুতরাং উত্তর হলো হ্যাঁ।
Google Translation: ডি কে মেটকাফ একজন আমেরিকান ফুটবল খেলোয়াড়। চাকা রুটে গাড়ি চালানো আমেরিকান ফুটবলের অংশ। তাহলে উত্তর হল হ্যাঁ।

Table 9: Example of **Google Translation**'s sub-par performance compared to **Manual Translation** when given the **Source** English text with the mistake underlined in pink (■).

D.2 Example on Basketball Context

Source: Ben Simmons is a basketball player. Calling for the screen is part of basketball. So the answer is yes.
Manual Translation: বেন সিমন্স একজন বাস্কেটবল খেলোয়াড়। স্ক্রিনের জন্য কল করা বাস্কেটবলের অংশ। সুতরাং উত্তর হলো হ্যাঁ।
Google Translation: বেন সিমন্স একজন বাস্কেটবল খেলোয়াড়। পর্দায় ডাকা বাস্কেটবলেরই অংশ। তাহলে উত্তর হল হ্যাঁ।

Table 10: Example of **Google Translation**'s sub-par performance on Basketball context with the mistake underlined in pink (■).

BanglaTalk: Towards Real-Time Speech Assistance for Bengali Regional Dialects

Jakir Hasan
Shahjalal University of Science and Technology, BD
jakirhasan718@gmail.com

Shubhashis Roy Dipta
University of Maryland, Baltimore County, USA
sroydip1@umbc.edu

Abstract

Real-time speech assistants are becoming increasingly popular for ensuring improved accessibility to information. Bengali, being a low-resource language with a high regional dialectal diversity, has seen limited progress in developing such systems. Existing systems are not optimized for real-time use and focus only on standard Bengali. In this work, we present **BanglaTalk**, the first real-time speech assistance system for Bengali regional dialects. BanglaTalk follows the client-server architecture and uses the Real-time Transport Protocol (RTP) to ensure low-latency communication. To address dialectal variation, we introduce a dialect-aware ASR system, **BRDialect**, developed by fine-tuning the IndicWav2Vec model in ten Bengali regional dialects. It outperforms the baseline ASR models by 12.41-33.98% on the RegSpeech12 dataset. Furthermore, BanglaTalk can operate at a low bandwidth of 24 kbps while maintaining an average end-to-end delay of 4.9 seconds. Low bandwidth usage and minimal end-to-end delay make the system both cost-effective and interactive for real-time use cases, enabling inclusive and accessible speech technology for the diverse community of Bengali speakers.[1]

1 Introduction

Conversational speech assistants (Dutsinma et al., 2022) have transformed human-computer interaction, making information more accessible. Widely adopted tools such as Alexa, Siri, and Cortana demonstrate the profound impact of real-time speech assistants on human lives (Hoy, 2018). However, while significant progress has been made for high-resource languages such as English, Mandarin, and French, such tools are still underdeveloped for the low-resource Bengali language. Bengali is a morphologically rich Indo-Aryan language (Islam et al., 2025), spoken by approximately 260 million people worldwide. It exhibits significant regional dialectal diversity, with variations in phonology, vocabulary, and syntax (Hasan et al., 2024b). This linguistic diversity poses a major challenge in building robust speech assistants.

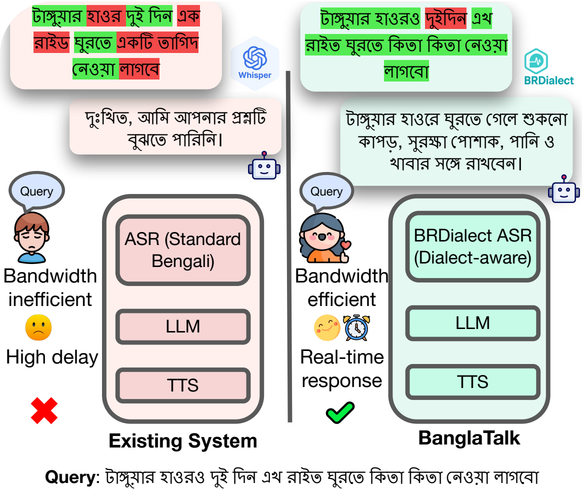

টাঙ্গুয়ার হাওর দুই দিন এক
রাইড ঘুরতে একটি তাগিদ
নেওয়া লাগবে
Whisper
দুঃখিত, আমি আপনার প্রশ্নটি বুঝতে পারিনি।
টাঙ্গুয়ার হাওরও দুইদিন এথ
রাইত ঘুরতে কিতা কিতা নেওয়া
লাগবো
BRDialect
টাঙ্গুয়ার হাওরে ঘুরতে গেলে শুকনো কাপড়, সুরক্ষা পোশাক, পানি ও খাবার সঙ্গে রাখবেন।
Query
Bandwidth inefficient
High delay
ASR (Standard Bengali)
LLM
TTS
Existing System
Query
Bandwidth efficient
Real-time response
BRDialect ASR (Dialect-aware)
LLM
TTS
BanglaTalk
Query: টাঙ্গুয়ার হাওরও দুই দিন এথ রাইত ঘুরতে কিতা কিতা নেওয়া লাগবো

Figure 1: Existing Bengali speech assistants (left) fail to understand queries in regional dialects due to reliance on standard Bengali ASR (incorrect transcriptions are shown in red). **BanglaTalk (right) successfully handles regional dialect queries through its dialect-aware ASR (BRDialect).** It is bandwidth efficient and operates in real-time due to the incorporation of the Real-Time Transport Protocol.

Automatic Speech Recognition (ASR) is a key component of speech assistant systems. Existing ASR systems are developed primarily for standard Bengali (Saha et al., 2021; Rakib et al., 2023b), and their performance is significantly degraded in regional dialects. As a result, existing speech assistant systems that integrate such ASR cannot support regional dialectal communication (Hasan et al., 2021; Arnab et al., 2023). Moreover, real-time deployment requires not only dialectal robustness, but also minimal end-to-end delay and efficient bandwidth usage. Previous works lack dialect-aware

[1] https://github.com/Jak57/BanglaTalk

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 44–60
December 23, 2025 ©2025 Association for Computational Linguistics

ASR, systematic analysis of delay minimization techniques, bandwidth efficiency, and real-time communication.

In this work, we introduce BanglaTalk, the first real-time conversational speech assistant for Bengali regional dialects. BanglaTalk adopts a client-server architecture and incorporates the Real-time Transport Protocol (RTP) (Schulzrinne et al., 2003) to achieve low-latency communication. Robust audio encoding enables operation at 24 kbps (kilobits per second). As illustrated in Fig. 1, while existing speech assistants fail in interpreting regional dialectal queries, BanglaTalk transcribes them accurately through the dialect-aware ASR system. It responds to queries effectively and interactively in real-time using low bandwidth.

The BanglaTalk client integrates lightweight audio processing modules, including noise cancellation, dynamic range compression, and audio encoding. On the server side, a dialect-aware ASR system, a voice activity detector (VAD), a natural-sounding Text-to-Speech (TTS) system, and audio encoding modules form a complete pipeline for real-time speech assistance. Central to this system is BRDialect, a dialect-aware ASR model fine-tuned on ten Bengali regional dialects. BRDialect outperforms the baseline Whisper (Tugstugi, 2023) and IndicWav2Vec (Javed et al., 2022) models, achieving a word error rate of 74.1% and character error rate of 40.6% on the RegSpeech12 (Hassan et al., 2025) dataset.

Additionally, the integrated VITS (Kim et al., 2021) TTS model produces natural-sounding speech with a high mean opinion score (MOS) of 4.49, enhancing the user experience. With an average end-to-end delay of 4.9 seconds, BanglaTalk enables interactive real-time communication between the user and the speech assistant. This system will significantly impact the lives of Bengali speakers due to its dialect-aware ASR, low bandwidth usage, and real-time performance.

In summary, our main contributions are:

- We introduce BanglaTalk, the first real-time, bandwidth-efficient Bengali speech assistant designed to support regional dialects through a client-server architecture.

- We develop BRDialect, a dialect-aware ASR system that substantially outperforms existing ASR models on the RegSpeech12 dataset spanning twelve regions of Bangladesh.

- We provide a comprehensive analysis of audio processing latency, bandwidth usage, end-to-end delay, and generated speech quality, demonstrating the robustness of BanglaTalk for real-time, dialect-aware communication.

2 Methodology

BanglaTalk follows a client-server architecture, with lightweight audio processing on the client and computationally intensive tasks on a centralized server. The overall pipeline is illustrated in Fig. 2.

2.1 BanglaTalk Client

The client is responsible for capturing, processing, and transmitting audio to the server. As shown in Fig. 2 (left), its main modules include audio capture, dynamic range compression, noise suppression, encoding, and transmission.

Audio Capture The client captures audio in 20-ms (milliseconds) frames at a sample rate of 16 kHz (kilohertz). Each frame contains 320 samples in 16-bit PCM little-endian format (Dobson, 2000). Although the Opus codec (Valin et al., 2016) supports multiple sample rates (e.g., 8-48 kHz), we fix the sample rate at 16 kHz to align with the ASR and VAD modules. Opus allows frame durations of 2.5-100 ms. A frame duration of 20 ms (50 RTP packets per second) offers a balance between packet size and loss rate in real-time communication.

Dynamic Range Compression (DRC) Speech captured from the microphone often includes soft and excessively loud signals. Compressing the dynamic range helps maintain a consistent audio level, enhancing performance (Giannoulis et al., 2012). We develop the dynamic range compression algorithm described in Alg. 1 and apply compression only to the loud audio segments. Whenever the decibel level of a normalized audio sample exceeds -10 dBFS (decibels relative to full scale), we apply a compression ratio of 2:1. Samples outside this threshold remain unchanged.

RNNoise Cancellation Noise poses a major issue in audio communication. Background and foreground noise can be picked up by the microphone and transmitted to the network, degrading the overall performance of the system. To mitigate this, we perform noise suppression with RNNoise (Valin, 2018). It is a lightweight neural network-based denoiser capable of real-time operation.

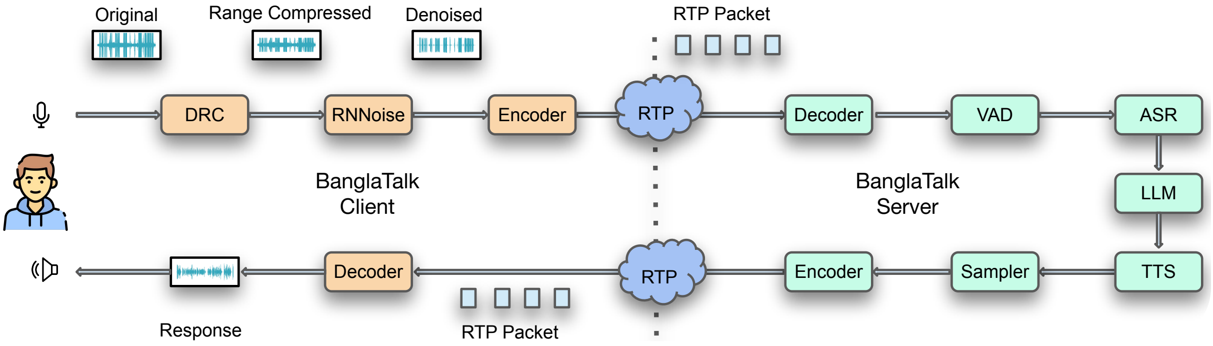


Figure 2: Client (left) and server-side (right) processing pipelines of the BanglaTalk System.

RNNoise is trained to remove noise from an audio frame of duration 10 ms at a 48 kHz sample rate. Since every captured audio frame duration is 20 ms at a 16 kHz sample rate, we apply audio segmentation, upsampling, and downsampling during noise suppression. Alg. 2 shows the pseudocode for upsampling. Linear interpolation (Xu and Xu, 2022) is used in upsampling from 16 to 48 kHz due to minimal computational overhead. After denoising, the audio is downsampled to 16 kHz using Alg. 3. Specifically, we skip intermediate sample values – from every three consecutive audio samples, the first one is retained and the remaining two are discarded. This simple technique is computationally efficient for downsampling.

Encoding with Opus Codec Opus is a high-quality audio codec for interactive speech and music transmission over the Internet (Valin et al., 2016). It is widely used in VoIP (Voice over IP) applications (Sundvall, 2014) due to the low latency processing and error concealment. Each audio frame contains a total of 320 samples (640 bytes). Without compression, audio frames are expensive to transmit over the Internet due to high bandwidth consumption. To mitigate this, each audio frame is encoded using the Opus codec with a low bitrate of 24 kbps. This ensures low bandwidth usage, which is a critical factor for greater accessibility. The incoming audio frames from the server are decoded with the same codec.

Packetization and Transmission Each encoded audio frame is encapsulated in an RTP packet following the Real-time Transport Protocol (Group et al., 1996). The first 12 bytes of each packet contain the header, and the remaining bytes contain the Opus-encoded payload. App. B describes the structure of the RTP packet. These packets are sent at 20 ms intervals to the server's public IP and the RTP packet receiver port (Postel, 1980).

2.2 BanglaTalk Server

The server is responsible for receiving the audio stream from the client and generating an appropriate audio response. The overall server-side processing pipeline is illustrated in Fig. 2 (right). To function effectively, the server employs several interconnected modules. First, the incoming audio frames are received and decoded using the Opus decoder. Next, voice activity detection is performed to identify speech segments. When a complete user query is detected, the corresponding speech segment is transcribed into text. This text is then processed by a large language model (LLM), which generates a suitable response. The response is subsequently converted into speech using a TTS system. Finally, the speech is resampled and the resulting audio frames are encoded with the Opus codec. Following the RTP protocol, the created RTP packets are transmitted to the client. The detailed workflow of these modules is described below.

RTP Packet Parser The server receives audio data from the client in the form of RTP packets. Each packet is parsed to extract header information, encoded data length, and encoded audio data in byte format.

Decoding with Opus Codec The extracted encoded audio is decoded using the Opus codec. Based on the encoded data length and audio bytes, the decoder reconstructs audio frames of 20 ms duration, corresponding to 320 samples at a 16 kHz sample rate.

Voice Activity Detection (VAD) Voice activity detection is a crucial component of real-time communication, as it identifies speech segments while discarding non-speech portions. This prevents unnecessary downstream processing, thus reducing la-

Algorithm 1 DRC compresses the amplitude of samples exceeding a -10 dBFS threshold by applying a 2:1 compression ratio. It leaves the quieter samples unchanged.

```
Require: Frame $x \in \mathbb{Z}^N$, threshold $\tau = -10$ dBFS, ratio $r = 2$
Ensure: Compressed frame $y \in \mathbb{Z}^N$
$y \leftarrow x$
for $i \leftarrow 1$ to $N$ do
    $s \leftarrow y_i$
    if $s = 0$ then
        continue
    end if
    $d \leftarrow 20 \cdot \log_{10}\big(|s|/32768\big)$
    if $d \leq \tau$ then
        continue
    end if
    $d' \leftarrow \tau + (d - \tau)/r$
    $\sigma \leftarrow \mathrm{sign}(s)$
    $s' \leftarrow \left\lfloor 10^{d'/20} \cdot \sigma \cdot 32767 \right\rfloor$
    $y_i \leftarrow s'$
end for
return $y$
```

tency and improving overall efficiency. We use the Silero VAD (Team, 2024) in the streaming mode, which is specifically designed for real-time applications. It works with a frame duration of 32 ms (512 samples) at a sample rate of 16 kHz and determines whether each audio frame marks the beginning, end of a speech segment, or none. Only the detected speech segment corresponding to the user query is forwarded to the ASR system.

Automatic Speech Recognition (ASR) We train a Wav2Vec2-based model (Baevski et al., 2020) using speech data from ten regional dialects from the Ben10 dataset (Humayun et al., 2024). For data processing, we follow Hasan et al. (2024a) and fine-tune the pre-trained IndicWav2Vec (Javed et al., 2022) for Bengali on that processed dataset. To evaluate the performance of our trained ASR model, we use the RegSpeech12 (Hassan et al., 2025) test set (Ben10 test set is not publicly available). A detailed description and analysis of these datasets are provided in App. C.1 and App. C.2.

End of Query Detection Accurate detection of the end of a user query is essential for real-time speech assistance systems (Liang et al., 2023). We have defined the end of query as a silence segment lasting at least 1.2 seconds. For silence detection, Silero VAD is utilized. Once an end-of-query is detected, the speech segment is passed to the ASR system to generate the transcription. The resulting query is then forwarded to the LLM, which produces the system's response.

Generating Response Using LLM Large Language Models (LLMs) are crucial for generating responses to user queries (Dam et al., 2024). To maintain coherent communication, responses are generated for valid queries, while invalid queries are discarded. We employ `GPT-4.1-nano` as the chat model, with the prompt template presented in App. D. To minimize latency, we use streaming mode, which delivers responses incrementally rather than waiting for a full response. The streamed text is segmented based on Bengali punctuation and forwarded to the TTS system.

Text-to-Speech (TTS) Several TTS models are available for Bengali (Raju et al., 2019). We experiment with MMS-TTS-Ben (Pratap et al., 2024) and two variants of VITS-Bengali (male and female voices) (Hossen, 2023). These models are selected because of their minimal processing delay. MMS-TTS-Ben produces speech at a 16 kHz sample rate, while VITS-Bengali outputs at 22.05 kHz. For system compatibility, the VITS-generated speech is resampled to 16 kHz.

Network Transmission The processed audio is segmented into 20 ms frames and encoded with the Opus codec at a bitrate of 24 kbps. Each RTP packet is constructed with a 12-byte header, followed by encoded audio data. The RTP packets are transmitted over the Internet to the client's public IP and port at 20 ms intervals, ensuring synchronized real-time playback.

3 Result & Discussion

3.1 Implementation Details

Since the system is intended for deployment across a diverse population in Bangladesh, we prioritize computational efficiency on the client side to ensure accessibility across devices with varying hardware capabilities. In contrast, the server must be sufficiently powerful to handle audio processing, response generation, and real-time communication with minimal latency. Accordingly, our experiments were conducted with an Intel Core i7 CPU (without GPU) as the client and a server equipped

Model	WER ↓	CER ↓
Whisper-medium-Bengali	0.846	0.562
IndicWav2Vec-Bengali	0.897	0.615
BRDialect	**0.741**	**0.406**

Table 1: Performance of ASR systems on the test set of the RegSpeech12 dataset, covering twelve Bengali regional dialects.

with an NVIDIA GeForce RTX 4090 GPU for efficient processing.

3.2 Evaluating BRDialect

To evaluate the performance of our dialect-aware ASR system, **BRDialect**, we use the test set from the RegSpeech12 (Hassan et al., 2025) dataset. It includes dialects from twelve regions of Bangladesh – Rangpur, Sylhet, Chittagong, Noakhali, Narail, Kishoreganj, Barishal, Habiganj, Comilla, Tangail, Sandwip, and Narsingdi, totaling 2132 audio files.

For evaluation, we report the Word Error Rate (WER) and Character Error Rate (CER), following the formulas described in App. E.1. To refine ASR predictions, we apply beam search decoding with a 5-gram KenLM language model (Heafield, 2011). We also investigated the impact of the preprocessing and postprocessing steps, including noise cancellation, normalization, and punctuation removal, as these factors significantly affect the overall performance of the ASR.

As shown in Table 1, **BRDialect** outperforms baseline models in both WER and CER. Specifically, it achieves the lowest WER of 0.741 and CER of 0.406 when decoded with a 5-gram KenLM model, combined with Unicode normalization and punctuation removal, but without noise cancellation.

BRDialect consistently outperforms `Whisper-medium-Bengali` (Tugstugi, 2023) and `IndicWav2Vec-Bengali` (Javed et al., 2022), achieving a 12.41–17.39% relative improvement in WER and a 27.77–33.98% improvement in CER. The comparatively poor performance of the baseline models suggests that dialectal variation is not adequately addressed during their training, limiting their suitability for regional speech-to-text tasks. BRDialect highlights the importance of dialect-aware fine-tuning for building a robust ASR system.

Processing	WER ↓	CER ↓
Noise Cancellation	0.876	0.497
No Noise Cancellation	0.865	0.452
5-gram KenLM Decoding	0.827	0.442
Unicode Normalization	0.796	0.420
Punctuation Removal	**0.741**	**0.406**

Table 2: Impact of different types of processing on the performance of the BRDialect ASR system. Processing pipelines are combined from top to bottom consecutively for the group without noise cancellation.

Performance across Regions We further evaluate BRDialect across individual regions. As shown in Fig. 3, the ASR system performs well across most regions, with WER below 70% in seven out of the twelve regions. The lowest WER, 0.438, is achieved for the Comilla region. Although the Comilla dialect is not included in the training data of BRDialect, its low WER demonstrates the model's strong generalization capability to unseen dialects.

3.2.1 Ablation Study

We analyze the effect of different preprocessing and postprocessing techniques on ASR performance. Table 2 summarizes the improvements observed with BRDialect.

Impact of Noise Cancellation We experiment with denoising the input audio before transcribing. The experimental results show that denoising with RNNoise slightly increases WER by 1.27%. Aggressive denoising can remove speech cues necessary for the accurate transcription of regional dialects. For the remaining processing steps, we keep the original audio without denoising and combine processing pipelines.

Impact of 5-gram KenLM Decoding Integrating a 5-gram KenLM model during decoding improves WER from 0.865 to 0.827, a 4.39% reduction. This confirms the value of training a KenLM language model with a Bengali regional text corpus for robust ASR performance (Rakib et al., 2023b).

Impact of Unicode Normalization Normalizing text further reduces WER to 0.796. This step is crucial since many Bengali characters can be represented in multiple ways, causing transcription inconsistencies. Normalizing text with the `BnUnicodeNormalizer` (Ansary et al., 2023) en-

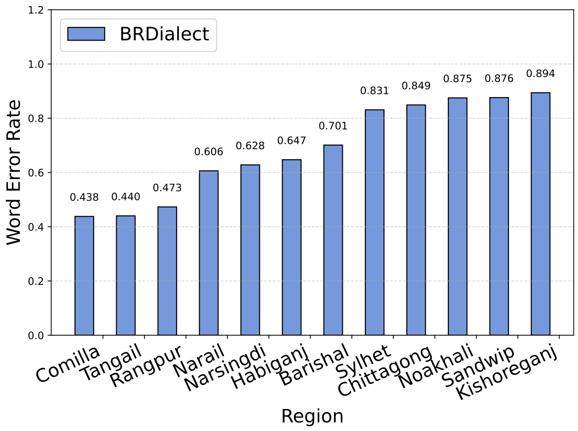


Figure 3: Regionwise word error rate distribution of the test set of the RegSpeech12 dataset. Transcriptions are generated using the BRDialect ASR system.

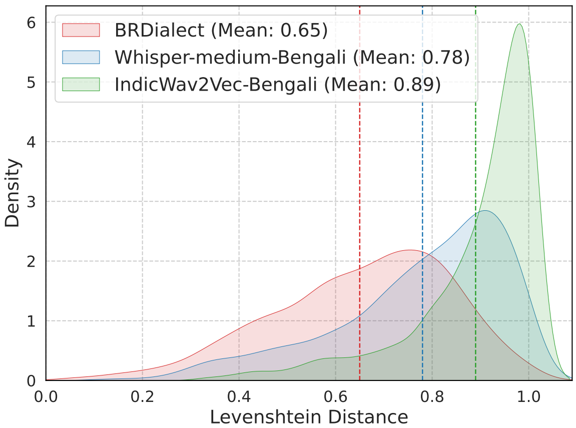


Figure 4: Distribution of Levenshtein distance for the best processing settings - without noise cancellation, Bangla unicode normalization, and punctuation removal by three ASR systems on the RegSpeech12 dataset.

sures uniform representation, improving ASR performance.

Impact of Punctuation Removal Since baseline ASR models do not generate punctuation, we remove punctuation from the RegSpeech12 transcriptions to ensure a fair comparison. Combined with previous steps, this yields the lowest WER of 0.741 and CER of 0.406. A detailed analysis of the processing configurations and their impact on the BRDialect ASR is provided in App. E.2.

Impact of High WER A Word Error Rate (WER) of 74.1% is relatively high for a general-purpose ASR system. However, dialectal speech recognition is inherently more challenging than the standard ASR task. The poor performance of the baseline models (`Whisper-medium-Bengali` and `IndicWav2Vec-Bengali`) validates this difficulty. In this context, BRDialect achieves a relative improvement of 12.41-33.98% over the baseline models, representing a substantial achievement.

Within the BanglaTalk system, the integrated large language model (`GPT-4.1-nano`) effectively compensates for minor transcription errors. Leveraging its robust contextual understanding, the LLM can infer user intent even from noisy or imperfect ASR outputs, as illustrated in Fig. 10. Most observed errors involve minor substitutions that do not significantly affect the intended meaning.

Furthermore, explicitly prompting the LLM to interpret dialectal queries (Fig. 7) enhances the system's ability to generate appropriate responses. The combination of a dialect-aware ASR model (BRDialect) and the powerful LLM (`GPT-4.1-nano`) ensures that the BanglaTalk system remains usable and effective despite relatively imperfect transcriptions.

3.2.2 Levenshtein Distance Analysis

Fig. 4 illustrates the distribution of normalized Levenshtein distance between the ground-truth transcriptions and the outputs of the evaluated ASR systems under the best processing configuration – No noise cancellation, Unicode normalization, and punctuation removal. For BRDialect, transcription quality is further refined during decoding using our trained 5-gram KenLM language model. A lower Levenshtein distance indicates higher transcription accuracy.

Among the models, BRDialect achieves the lowest mean distance of 0.65, demonstrating strong alignment with the reference transcriptions. `Whisper-medium-Bengali` performs moderately with a mean distance of 0.78, while `IndicWav2Vec-Bengali` consistently underperforms. The distribution of `IndicWav2Vec-Bengali` peaks sharply at 1, with the highest mean distance 0.89, highlighting substantial deviation from the ground truth.

In contrast, BRDialect exhibits a broader distribution, reflecting variability in performance – some utterances are transcribed with high accuracy, while others are less. It achieves a measurable improvement of 16.67-26.97% compared to the baseline models, highlighting its effectiveness in handling dialectal diversity.

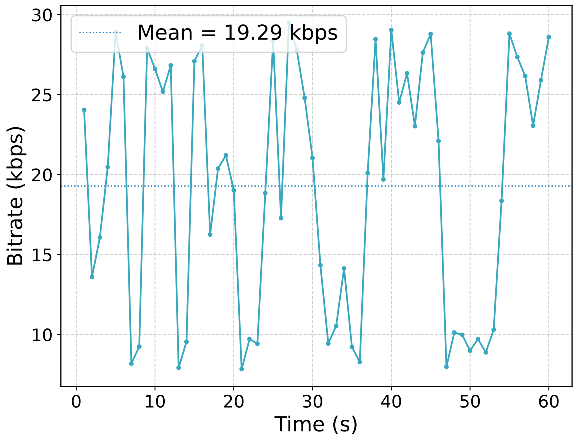


Figure 5: Uploading bitrate on the client side for a duration of one minute.

Module	Process Time (ms)
DRC	1.31
RNNoise	6.51
Encoding	**0.56**
Total	8.38

Table 3: Average processing times of a single audio frame of duration 20 ms.

3.3 Evaluating BanglaTalk

Low Latency Audio Processing On the client side, each captured audio frame undergoes three sequential processing steps – dynamic range compression, noise cancellation, and encoding with the Opus codec. The average processing time for an audio frame of duration 20 ms, is reported in Table 3.

Among these, the encoding with the Opus codec is the most efficient, requiring only 0.56 ms per frame. This is consistent with the codec's design, which targets real-time, low-latency audio applications (Valin et al., 2016). Similarly, the DRC module introduces minimal computational overhead, averaging 1.31 ms per frame. In contrast, the noise cancellation with RNNoise introduces the highest processing time of 6.51 ms.

RNNoise applies a lightweight neural network for speech enhancement, making it more resource-intensive than traditional signal processing methods. Additionally, RNNoise operates exclusively at a 48 kHz sample rate, while the BanglaTalk system relies on 16 kHz to ensure compatibility with ASR and VAD modules (see §2.1). As a result, each frame must first be upsampled from 16 kHz to 48 kHz prior to noise cancellation and subsequently downsampled back to 16 kHz. These resampling operations introduce an additional delay of 1.40 ms for upsampling and 0.34 ms for downsampling. Compared to other ML-based noise cancellation systems (Cha et al., 2023), RNNoise provides an excellent balance of speech quality and efficiency, making it highly suitable for real-time speech applications (Valin, 2018).

Despite this, the total processing time (8.38 ms) remains well below the 20 ms frame duration, ensuring that all processing completes before the arrival of the next frame. This guarantees uninterrupted, real-time streaming without additional latency.

Adaptive Bitrate Control In real-time assistive speech technologies, both end-to-end latency and bandwidth efficiency are crucial factors (Kaur et al., 2019). Bandwidth efficiency is particularly important for underserved populations, such as those in rural areas of Bangladesh, where high-speed or expensive Internet connections are often unavailable.

To address this challenge, our system prioritizes bandwidth efficiency without compromising performance. Audio streams are encoded at a target bitrate of 24 kbps, significantly reducing bandwidth usage in the upload and download streams during client-server communication. Fig. 5 shows the upload bitrate of client-side for a one-minute audio stream.

We employ the variable bitrate (VBR) mode of the Opus codec, which dynamically adjusts the bitrate based on the characteristics of the audio signal – allocating more bitrate to speech segments and conserving bitrate during silence. In the example shown in Fig. 5, the average bitrate over one minute is only 19.29 kbps. This adaptive bitrate usage makes the system more accessible to users in economically disadvantaged regions (Osuagwu et al., 2013), who might otherwise be unable to benefit from assistive speech technologies.

Although applying RNNoise noise cancellation slightly increases the ASR system's Word Error Rate (WER) by 1.27% (Table 2), it is retained in the client application due to its substantial benefit in reducing bandwidth usage. By removing background and foreground noise, the audio signal becomes more compressible. Without noise cancellation, the average upload bitrate of the audio (as shown in Fig. 5) rises from 19.29 kbps to 23.6 kbps – an 18.26% increase that is inefficient for client-side transmission. Given that the large language model (GPT-4.1-nano) demonstrates strong robustness in understanding minor transcription errors and generating accurate responses (Fig. 10), the advantages of incorporating RNNoise into the client application outweigh its modest negative impact on ASR performance.

End-to-End Latency For real-time assistive speech systems, minimizing the delay between the end of a user's query and the beginning of the system's audio response is essential to ensure natural interactivity. The BanglaTalk system is designed with this principle in mind, introducing minimal overhead in both client and server-side processing.

We conduct rigorous testing and time analysis to measure the BanglaTalk system's end-to-end delay. As detailed in App. E.3, the BRDialect ASR module of the BanglaTalk system introduces only a small delay, making it well-suited for real-time applications. The TTS systems evaluated in our study also exhibit low processing delay. App. F presents a detailed evaluation of TTS systems. Among them, the VITS-Bengali (male variant) is the best-performing model, achieving a high Mean Opinion Score (MOS) of 4.49 on the subset of the BanSpeech (Samin et al., 2024) dataset.

To quantify the overall end-to-end delay of the BanglaTalk system, we simulate ten user queries in a conversation setting and measure the time elapsed from the end of the user queries to the start of the system's response. As shown in App. G, the system achieves an average end-to-end delay of 4.9 seconds. This latency is acceptable for real-time assistive applications, ensuring smooth interactivity and an enhanced user experience.

Preliminary User Study To evaluate the real-world usability of the BanglaTalk system, we conduct a limited user study involving four native speakers of the Sylhet and Mymensingh dialects. The Sylhet dialect is selected due to its relative high WER (0.831), while the Mymensingh dialect is included to assess the generalizability of BRDialect to unseen dialects, as it is not part of the training set. Two native speakers from each region interact with BanglaTalk on general information and everyday task queries. Participants rate their interaction experience on a 1-5 scale, where 1 indicates a poor experience and 5 indicates an excellent experience. Table 9 summarizes the results of five queries per user. Overall, BanglaTalk achieves a mean rating of 3.62, indicating a generally positive user experience.

As illustrated in Fig. 10, users from both regions interact successfully with the system despite minor transcription errors. Notably, users from the Mymensigh region, whose dialect is not included in BRDialect's training data, report a mean satisfaction rating of 3.73, further demonstrating the model's strong generalization capability to unseen dialects. Participants also note that, despite their strong accent in regional dialects, BanglaTalk responds accurately and naturally, providing an interactive and effective speech-based experience.

4 Related Work

4.1 Bengali Speech Assistants

Several speech assistant systems have been developed for the Bengali language in recent years. The ALAPI system (Hasan et al., 2021) introduces an open-domain Bengali conversational agent that processes recorded audio queries from users and generates response audio using AI techniques alongside a custom-built database. Shohojogi (Arnab et al., 2023), designed for the banking sector, provides voice-based customer support in Bengali. It integrates Wav2Vec2-based ASR (Baevski et al., 2020), query summarization, Google Text-to-Speech (gTTS), and a doc2vec model to retrieve relevant information for responses. Adrisya Sahayak (Sultan et al., 2021) presents a desktop-based virtual speech assistant for visually impaired Bengali speakers, supporting computer operations, peripheral devices, and home appliance control.

In the healthcare domain, a voice-enabled Artificial Conversational Entity has been developed to automate service interactions in Bengali (Pranto et al., 2021). This system leverages a domain-specific database and similarity-matching strategies to generate responses to user queries. Extending beyond monolingual assistants, Disha (Ullah et al., 2024) is a humanoid virtual assistant capable of interacting in both Bengali and English. It can address financial queries and perform real-time transactions. Beyond general-purpose applications, Bengali voice assistants have also been deployed in specialized domains such as providing information on metro rail services in Bangladesh (Rahman et al., 2024), assisting farmers with agricultural queries (Divakar et al., 2021), and offering accessible feminine healthcare support for marginalized women (Puja et al., 2024).

4.2 English Speech Assistants

Voice assistants such as Amazon Alexa, Apple Siri, Microsoft Cortana, and Google Assistant have become integral to modern human-computer interaction (López et al., 2017). These systems, powered by artificial intelligence, are increasingly influencing daily life and societal practices (Subhash et al., 2020). Their development has been driven by advances in both signal processing and machine learning, which form the technological foundation of voice-based interaction (Haeb-Umbach et al., 2019).

Beyond their technical design, researchers have examined the broader impact of these systems. For example, Flavián et al. (2023) demonstrated that, compared to text-based recommendations, voice-based recommendations delivered by assistants have a stronger influence on consumer decision-making. User studies also indicate that voice assistants are most frequently employed for activities such as music, information search, and smart home (IoT) control (Ammari et al., 2019). Furthermore, their role has extended into education and healthcare contexts, where they support learning environments and assist older adults in managing everyday activities (Terzopoulos and Satratzemi, 2020; Oewel et al., 2023).

4.3 Automatic Speech Recognition in Bengali

Research on Automatic Speech Recognition (ASR) in Bengali has been addressed through both system development and survey-driven studies (Mridha et al., 2022; Tasnia et al., 2023; Sultana et al., 2021). The availability of large-scale datasets, such as Bengali Common Voice (Alam et al., 2022), OOD-Speech (Rakib et al., 2023a), and RegSpeech12 (Hassan et al., 2025), has created significant opportunities for advancing Bengali ASR systems. Continuous Bengali ASR has been benchmarked on the SHRUTI corpus using DNN-HMM and GMM-HMM-based models (Al Amin et al., 2019), achieving relatively low error rates. Leveraging CNN-RNN architectures, Saha et al. (2021) developed a gender- and speaker-independent ASR system. In Bangla-Wave (Rakib et al., 2023b), the integration of an n-gram language model is shown to yield notable performance improvements. Furthermore, comparative analysis demonstrates that the Wav2Vec-BERT model outperforms Whisper on the Bengali Common Voice dataset (Ridoy et al., 2025).

5 Conclusion & Future Work

In this work, we present BanglaTalk, the first real-time, end-to-end conversational speech assistant designed specifically for Bengali regional dialects. The system integrates both client and server applications, with low-latency audio signal processing implemented on both ends. To ensure efficient network transmission, speech data is transmitted at a low bandwidth of 24 kbps, making the system accessible to a broad range of users. Our developed BRDialect ASR system, integrated into the BanglaTalk pipeline, effectively transcribes Bengali regional dialects, achieving an overall word error rate of 74.1% and a character error rate of 40.6% on the RegSpeech12 dataset, which spans 12 regions of Bangladesh. For speech synthesis, the VITS-Bengali TTS model (male) incorporated into the system attains a mean opinion score (MOS) of 4.49 on a subset of the BanSpeech dataset, enhancing the naturalness and human-like quality of the generated voice. Furthermore, the system maintains a low end-to-end delay of 4.9 seconds, ensuring a highly interactive user experience. These performance metrics demonstrate the effectiveness of the system for real-time communication.

Limitations

The BRDialect ASR system is trained on regional speech data covering ten regions of Bangladesh. Regions not included in the training data may experience less accurate transcriptions when using the BanglaTalk system. Incorporating speech data from the remaining regions of Bangladesh is expected to significantly improve the performance of the BRDialect ASR system. Although BanglaTalk reduces end-to-end delay and is highly interactive, several limitations remain:

- User interruption: In the current system, the assistant continues speaking until the utterance is completed. Adding the capability to handle user interruptions would make conversations more natural and interactive.
- Speaker verification: The system does not verify the speaker. If another person speaks during the user's conversation, their speech is transcribed, which negatively affects performance. Incorporating speaker verification would mitigate this issue.
- Concurrent conversations: At present, only one conversation can be executed at a time.

Supporting multiple concurrent conversations would increase system availability and usability.

- User study coverage: Feedback has so far been collected from a limited set of regional speakers. A broader user study covering all regions of Bangladesh would provide deeper insights into system performance, user acceptance, and overall impact.

References

Md Alif Al Amin, Md Towhidul Islam, Shafkat Kibria, and Mohammad Shahidur Rahman. 2019. Continuous bengali speech recognition based on deep neural network. In *2019 international conference on electrical, computer and communication engineering (ECCE)*, pages 1–6. IEEE.

Samiul Alam, Asif Sushmit, Zaowad Abdullah, Shahrin Nakkhatra, MD Ansary, Syed Mobassir Hossen, Sazia Morshed Mehnaz, Tahsin Reasat, and Ahmed Imtiaz Humayun. 2022. Bengali common voice speech dataset for automatic speech recognition. *arXiv preprint arXiv:2206.14053*.

Tawfiq Ammari, Jofish Kaye, Janice Y Tsai, and Frank Bentley. 2019. Music, search, and iot: How people (really) use voice assistants. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(3):1–28.

Nazmuddoha Ansary, Quazi Adibur Rahman Adib, Tahsin Reasat, Asif Shahriyar Sushmit, Ahmed Imtiaz Humayun, Sazia Mehnaz, Kanij Fatema, Mohammad Mamun Or Rashid, and Farig Sadeque. 2023. Unicode normalization and grapheme parsing of indic languages. *arXiv preprint arXiv:2306.01743*.

Kabir Abdur Rahman Arnab, Ashiqual Hossain, Istihad Nabi, and Muhammad Iqbal Hossain. 2023. *Shohojogi: An automated voice chat system in the bangla language for the banking system*. Ph.D. thesis, Ph. D. Dissertation. https://doi. org/10. 13140/RG. 2.2. 34757.22240/1.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

Young-Jin Cha, Alireza Mostafavi, and Sukhpreet S Benipal. 2023. Dnoisenet: Deep learning-based feedback active noise control in various noisy environments. *Engineering Applications of Artificial Intelligence*, 121:105971.

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. 2024. A complete survey on llm-based ai chatbots. *arXiv preprint arXiv:2406.16937*.

MS Divakar, Vimal Kumar, Martina Jaincy DE, RA Kalpana, Sanjai Kumar RM, and 1 others. 2021. Farmer's assistant using ai voice bot. In *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, pages 527–531. IEEE.

Richard W Dobson. 2000. Developments in audio file formats. In *ICMC*.

Faruk Lawal Ibrahim Dutsinma, Debajyoti Pal, Suree Funilkul, and Jonathan H Chan. 2022. A systematic review of voice assistant usability: An iso 9241–11 approach. *SN computer science*, 3(4):267.

Carlos Flavián, Khaoula Akdim, and Luis V Casaló. 2023. Effects of voice assistant recommendations on consumer behavior. *Psychology & Marketing*, 40(2):328–346.

Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss. 2012. Digital dynamic range compressor design—a tutorial and analysis. *Journal of the Audio Engineering Society*, 60(6):399–408.

Audio-Video Transport Working Group, H Schulzrinne, S Casner, R Frederick, V Jacobson, and 1 others. 1996. Rfc1889: Rtp: A transport protocol for real-time applications.

Reinhold Haeb-Umbach, Shinji Watanabe, Tomohiro Nakatani, Michiel Bacchiani, Bjorn Hoffmeister, Michael L Seltzer, Heiga Zen, and Mehrez Souden. 2019. Speech processing for digital home assistants: Combining signal processing with deep-learning techniques. *IEEE Signal processing magazine*, 36(6):111–124.

Jakir Hasan, Md. Ataullha Saim, and Radeen Mostafa. 2024a. Improving bengali asr for regional dialects with indicwav2vec: A competition approach. Preprint, ResearchGate.

Md Mehedi Hasan, Auronno Roy, and Md Tariq Hasan. 2021. Alapi: An automated voice chat system in bangla language. In *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, pages 1–4. IEEE.

Md Nahid Hasan, Raiyan Azim, and Sadia Sharmin. 2024b. Credibility analysis of robot speech based on bangla language dialect. In *2024 IEEE International Conference on Computing, Applications and Systems (COMPAS)*, pages 1–6. IEEE.

Md. Rezuwan Hassan, Azmol Hossain, Kanij Fatema, Rubayet Sabbir Faruque, Tanmoy Shome, Ruwad Naswan, Trina Chakraborty, Tawsif Tashwar Dipto, Md Foriduzzaman Zihad, Nazmuddoha Ansary, Asif Sushmit, Ahmed Imtiaz Humayun, Tahsin Reasat, Md Mehedi Hasan Shawon, Md. Golam Rabiul Alam, and Farig Sadeque. 2025. Regspeech12: A regional corpus of bengali spontaneous speech across dialects. Kaggle Dataset. Accessed: 2025-09-19.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197.

Mobassir Hossen. 2023. Comprehensive bangla tts. Kaggle dataset. Accessed: 2025-09-19.

Matthew B Hoy. 2018. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88.

Ahmed Imtiaz Humayun, farigys, Mohaymen Ul Anam, Rubayet Sabbir Faruque, S. M. Jishanul Islam, Sushmit, and Tahsin. 2024. Asr for regional dialects. Kaggle competition. Accessed: 2025-09-19.

Md Fuadul Islam, Jakir Hasan, Md Ashikul Islam, Prato Dewan, and M Shahidur Rahman. 2025. Banglalem: a transformer-based bangla lemmatizer with an enhanced dataset. *Systems and Soft Computing*, page 200244.

Tahir Javed, Sumanth Doddapaneni, Abhigyan Raman, Kaushal Santosh Bhogale, Gowtham Ramesh, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2022. Towards building asr systems for the next billion users. In *Proceedings of the aaai conference on artificial intelligence*, volume 36, pages 10813–10821.

Ravneet Kaur, Ravtej Singh Sandhu, Ayush Gera, Tarlochan Kaur, and Purva Gera. 2019. Intelligent voice bots for digital banking. In *Smart Systems and IoT: Innovations in Computing: Proceeding of SSIC 2019*, pages 401–408. Springer.

Jaehyeon Kim, Jungil Kong, and Juhee Son. 2021. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pages 5530–5540. PMLR.

Dawei Liang, Hang Su, Tarun Singh, Jay Mahadeokar, Shanil Puri, Jiedan Zhu, Edison Thomaz, and Mike Seltzer. 2023. Dynamic speech endpoint detection with regression targets. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Gustavo López, Luis Quesada, and Luis A Guerrero. 2017. Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In *International conference on applied human factors and ergonomics*, pages 241–250. Springer.

Muhammad Firoz Mridha, Abu Quwsar Ohi, Md Abdul Hamid, and Muhammad Mostafa Monowar. 2022. A study on the challenges and opportunities of speech recognition for bengali language. *Artificial Intelligence Review*, 55(4):3431–3455.

Bruna Oewel, Tawfiq Ammari, and Robin N Brewer. 2023. Voice assistant use in long-term care. In *Proceedings of the 5th International Conference on Conversational User Interfaces*, pages 1–10.

OE Osuagwu, S Okide, D Edebatu, and E Udoka. 2013. Low and expensive bandwidth remains key bottleneck for nigeria's internet diffusion: A proposal for a solution model. *West African Journal of Industrial and Academic Research*, 7(1):14–30.

Jon Postel. 1980. User datagram protocol. Technical report.

Shehan Irteza Pranto, Rahad Arman Nabid, Ahnaf Mozib Samin, Nabeel Mohammed, Farhana Sarker, Mohammad Nurul Huda, and Khondaker A Mamun. 2021. Human-robot interaction in bengali language for healthcare automation integrated with speaker recognition and artificial conversational entity. In *2021 3rd International Conference on Electrical & Electronic Engineering (ICEEE)*, pages 13–16. IEEE.

Vineel Pratap, Andros Tjandra, Bowen Shi, Paden Tomasello, Arun Babu, Sayani Kundu, Ali Elkahky, Zhaoheng Ni, Apoorv Vyas, Maryam Fazel-Zarandi, and 1 others. 2024. Scaling speech technology to 1,000+ languages. *Journal of Machine Learning Research*, 25(97):1–52.

Sreya Sanyal Puja, Nahian Noor Neha, Ofia Rahman Alif, Tarannaum Jahan Sultan, Md Golam Zel Asmaul Husna, Ishrat Jahan, and Jannatun Noor. 2024. Exploring the barriers to feminine healthcare access among marginalized women in bangladesh and facilitating access through a voice bot. *Heliyon*, 10(14).

MD Rahman, Adnan Alamgir, Shaheedul Haque Chowdhury, Maliha Mushtari, Wasim Anzum, and 1 others. 2024. *A comprehensive NLP-based voice assistant system for streamlined information retrieval in metro rail services of Bangladesh*. Ph.D. thesis, Brac University.

Rajan Saha Raju, Prithwiraj Bhattacharjee, Arif Ahmad, and Mohammad Shahidur Rahman. 2019. A bangla text-to-speech system using deep neural networks. In *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5. IEEE.

Fazle Rabbi Rakib, Souhardya Saha Dip, Samiul Alam, Nazia Tasnim, Md Istiak Hossain Shihab, Md Nazmuddoha Ansary, Syed Mobassir Hossen, Marsia Haque Meghla, Mamunur Mamun, Farig Sadeque, and 1 others. 2023a. Ood-speech: A large bengali speech recognition dataset for out-of-distribution benchmarking. *arXiv preprint arXiv:2305.09688*.

Mohammed Rakib, Md Ismail Hossain, Nabeel Mohammed, and Fuad Rahman. 2023b. Bangla-wave: Improving bangla automatic speech recognition utilizing n-gram language models. In *Proceedings of the 2023 12th International Conference on Software and Computer Applications*, pages 297–301.

Md Sazzadul Islam Ridoy, Sumi Akter, and Md Aminur Rahman. 2025. Adaptability of asr models on low-resource language: A comparative study of whisper and wav2vec-bert on bangla. *arXiv preprint arXiv:2507.01931*.

Srijoni Saha and 1 others. 2021. Development of a bangla speech to text conversion system using deep learning. In *2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–7. IEEE.

Ahnaf Mozib Samin, M Humayon Kobir, Md Mushtaq Shahriyar Rafee, M Firoz Ahmed, Mehedi Hasan, Partha Ghosh, Shafkat Kibria, and M Shahidur Rahman. 2024. Banspeech: A multi-domain bangla speech recognition benchmark toward robust performance in challenging conditions. *IEEE Access*, 12:34527–34538.

Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. 2003. Rtp: A transport protocol for real-time applications. Technical report.

Robert C Streijl, Stefan Winkler, and David S Hands. 2016. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227.

S Subhash, Prajwal N Srivatsa, S Siddesh, A Ullas, and B Santhosh. 2020. Artificial intelligence-based voice assistant. In *2020 Fourth world conference on smart trends in systems, security and sustainability (WorldS4)*, pages 593–596. IEEE.

Md Rakibuz Sultan, Md Moinul Hoque, Farah Ulfath Heeya, Iftiquar Ahmed, Md Redwanul Ferdouse, and Shikder Mejbah Ahmed Mubin. 2021. Adrisya sahayak: A bangla virtual assistant for visually impaired. In *2021 2nd international conference on robotics, electrical and signal processing techniques (ICREST)*, pages 597–602. IEEE.

Sadia Sultana, M Shahidur Rahman, and M Zafar Iqbal. 2021. Recent advancement in speech recognition for bangla: A survey. *International Journal of Advanced Computer Science and Applications*, 12(3).

Mika Sundvall. 2014. Opus audio codec in mobile networks.

Nabila Tasnia, Mahidul Islam, Mahi Shahriar Rony, Nishat Tanzim, Khan Md Hasib, and Mohammad Shafiul Alam. 2023. An overview of bengali speech recognition: Methods, challenges, and future direction. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0873–0878. IEEE.

Silero Team. 2024. Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier. `https://github.com/snakers4/silero-vad`.

George Terzopoulos and Maya Satratzemi. 2020. Voice assistants and smart speakers in everyday life and in education. *Informatics in Education*, 19(3):473–490.

Tugstugi. 2023. Bengali ai asr submission. Kaggle dataset. Accessed: 2025-09-19.

Md Rifat Ullah, Md Nafees Mahbub, Md Azizul Hakim, Yeasmin Sultana, and Iftakhar Alam. 2024. Disha: A bilingual humanoid virtual assistant. In *2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, pages 457–462. IEEE.

Jean-Marc Valin. 2018. A hybrid dsp/deep learning approach to real-time full-band speech enhancement. In *2018 IEEE 20th international workshop on multimedia signal processing (MMSP)*, pages 1–5. IEEE.

Jean-Marc Valin, Gregory Maxwell, Timothy B Terriberry, and Koen Vos. 2016. High-quality, low-delay music coding in the opus codec. *arXiv preprint arXiv:1602.04845*.

Yijie Xu and Runqi Xu. 2022. Research on interpolation and data fitting: Basis and applications. *arXiv preprint arXiv:2208.11825*.

A Algorithms

A.1 Upsampling

The upsampling algorithm described in Alg. 2 increases the audio sample rate by generating intermediate samples through linear interpolation.

Algorithm 2 Upsampling with linear interpolation increases the sample rate by inserting new sample values between consecutive samples. The intermediate values are calculated using the previous-current sample pair and the slope between them.

```
Require: Input frame $x \in \mathbb{Z}^N$, ratio $r$, previous value $p$
Ensure: Upsampled array $y \in \mathbb{Z}^{N \cdot r}$
$y \leftarrow$ array of zeros with length $N \cdot r$
$k \leftarrow 1$
for $i \leftarrow 1$ to $N$ do
    $cur \leftarrow x[i]$
    $\Delta \leftarrow (cur - p)/r$
    for $j \leftarrow 0$ to $r - 1$ do
        $v \leftarrow p + j \cdot \Delta$
        Clip $v$ into range $[-32768, 32767]$
        $y[k] \leftarrow \lfloor v \rceil$; $k \leftarrow k + 1$
    end for
    $p \leftarrow cur$
end for
return $y$
```

A.2 Downsampling

The downsampling algorithm described in Alg. 3 reduces the audio sample rate by discarding intermediate samples.

Algorithm 3 Downsampling by an integer factor r reduces the sample rate by keeping every $r - th$ sample from the speech signal.

```
Require: Input frame $x \in \mathbb{Z}^L$, ratio $r \in \mathbb{N}, r \geq 1$
Ensure: Downsampled array $y \in \mathbb{Z}^{\lfloor L/r \rfloor}$
$N \leftarrow \left\lfloor \frac{L}{r} \right\rfloor$
$y \leftarrow$ array of zeros with length $N$
$k \leftarrow 1$
for $i \leftarrow 1$ to $L$ step $r$ do
    $y[k] \leftarrow x[i]$;    $k \leftarrow k + 1$
    if $k > N$ then break
    end if
end for
return $y$
```

B The RTP Packet Structure

The structure of a standard RTP packet, which comprises a 12-byte header, is shown in Fig. 6.

C Dataset

C.1 Ben10 Dataset Analysis

The Ben10 dataset (Humayun et al., 2024) comprises speech recordings from 373 speakers across ten distinct regions of Bangladesh. The training set contains over 63 hours of audio sampled at 16 kHz. Table 4 presents the region-wise distribution of audio files within the training set. Since the transcripts of the Ben10 test set are not publicly available, we instead employ the test set of the RegSpeech12 dataset (Hassan et al., 2025) for evaluation in our study.

Training Set	
Region	**Audio File Count**
Barishal	796
Chittagong	1406
Habiganj	940
Kishoreganj	1638
Narail	1488
Narsingdi	1098
Rangpur	1037
Sandwip	1049
Sylhet	2903
Tangail	987
Total	**13342**

Table 4: Training set distribution of the Ben10 dataset

C.2 RegSpeech12 Dataset Analysis

The RegSpeech12 dataset (Hassan et al., 2025) is a spontaneous speech corpus encompassing twelve regional dialects of Bangladesh, comprising approximately 100 hours of speech data. In this study, we utilize the test split of the dataset, which contains around 10 hours of recordings. Table 5 presents the region-wise distribution of the test set.

D Prompt to LLM

To generate responses to spoken user queries transcribed by the dialect-aware ASR system, we provide the prompt illustrated in Fig. 7 to the GPT-4.1-nano model.

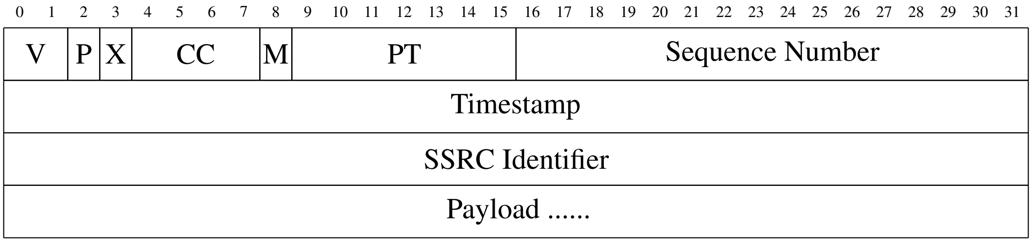


Figure 6: RTP packet structure with a 12-byte header followed by the audio payload.

Test Set	
Region	**Audio File Count**
Barishal	101
Chittagong	176
Comilla	32
Habiganj	117
Kishoreganj	205
Narail	186
Narsingdi	137
Noakhali	28
Rangpur	130
Sandwip	131
Sylhet	762
Tangail	127
Total	**2132**

Table 5: Test set distribution of the RegSpeech12 dataset

System Prompt: You are a helpful chatbot who understands Bengali regional dialects and only speaks standard Bengali. Please be concise and end every sentence with {|}.

User Prompt: Please generate a response for only the valid query. For an invalid query, print only a {$}. Here is the query in the Bengali regional dialect {user_query}.

Figure 7: Prompt for LLM to generate response for query in Bengali regional dialect.

E ASR System Evaluation

E.1 ASR Evaluation Metrics

Word Error Rate (WER) and Character Error Rate (CER) are the standard metrics for evaluating the performances of ASR systems. The formula for Word Error Rate is:

$$\text{WER} = \frac{S + D + I}{N} \times 100\%, \qquad (1)$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the total number of words in the reference transcription.

Similarly, the formula for Character Error Rate is:

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c} \times 100\%, \qquad (2)$$

where S_c is the number of substitutions, D_c is the number of deletions, and I_c is the number of insertions at the character level. N_c is the total number of characters in the reference transcription.

E.2 Impact of Processing on BRDialect ASR System

Processing pipelines play a critical role in shaping the performance of ASR systems. Table 6 presents the impact of 5-gram KenLM decoding, noise cancellation, text normalization, and punctuation removal on the BRDialect ASR system.

E.3 Processing Times of ASR Systems

The inference time for audio files varies across different ASR systems, as shown in Table 7. The Whisper model exhibits significantly slower processing compared to BRDialect and IndicWav2Vec. Although the IndicWav2Vec model is slightly faster than BRDialect, its overall performance is considerably lower. In contrast, the processing time of BRDialect falls within an acceptable range for real-time communication applications.

F TTS Systems Evaluation

In this study, we conduct extensive experiments with three open-source TTS systems, all of which are based on the VITS architecture (Kim et al., 2021): MMS-TTS-Ben (Pratap et al., 2024), and the VITS-Bengali Male and Female variants (Hossen, 2023). To evaluate the quality of the synthesized speech, we employ the mean opinion score (MOS) metric.

KD	NC	UN	PR	WER	CER
✓	×	×	×	0.827	0.442
✓	×	×	✓	0.796	0.420
✓	×	✓	×	0.781	0.430
✓	×	✓	✓	**0.741**	**0.406**
✓	✓	×	×	0.838	0.493
✓	✓	×	✓	0.813	0.474
✓	✓	✓	×	0.801	0.479
✓	✓	✓	✓	0.770	0.458
×	×	×	×	0.865	0.452
×	×	×	✓	0.834	0.444
×	×	✓	×	0.834	0.429
×	×	✓	✓	0.793	0.419
×	✓	×	×	0.876	0.497
×	✓	×	✓	0.853	0.487
×	✓	✓	×	0.853	0.478
×	✓	✓	✓	0.822	0.467

Table 6: Performance of the BRDialect ASR system on the processing settings of 5-gram KenLM Decoding (KD), Noise Cancellation (NC), Unicode Normalization (UN), and Punctuation Removal (PR).

For a robust comparison, we curate a diverse set of texts from the BanSpeech dataset (Samin et al., 2024), which contains audio-text pairs across thirteen categories. From each category, one representative text sample is selected, forming the test dataset, as detailed in App. F.1. Each text sample is synthesized using all three TTS models. To assess naturalness and perceived audio quality, an experienced human rater with expertise in audio signal processing independently rates each generated audio on a 1 to 5 scale, where 1 indicates very poor quality and 5 represents excellent quality (Streijl et al., 2016).

The results, summarized in Fig. 8, indicate that the VITS-Bengali Male model achieves the highest average MOS score of 4.49, producing speech that is perceived as highly natural and pleasant. The VITS-Bengali Female model achieves an average MOS of 4.40, which is also suitable for end-to-end speech assistant systems. In contrast, MMS-TTS-Ben performs the lowest, with an average MOS score of 3.66, approximately 22.7% lower than VITS-Bengali Male, indicating reduced suitability for end-to-end applications.

F.1 Text Samples from the BanSpeech Dataset

The BanSpeech dataset (Samin et al., 2024) comprises audio–text pairs spanning diverse categories. For evaluating the TTS systems, we use the text samples from this dataset, as illustrated in Fig. 9.

Audio	**Preprocessing Times (s) ↓**		
	Whisper	IndicWav2Vec	BRDialect
sylhet_1	3.07	0.65	**0.77**
sylhet_2	3.61	0.59	**1.00**
sylhet_3	3.99	1.36	**1.31**
sylhet_4	3.49	0.80	**0.97**

Table 7: Processing time analysis of four audio files with an average duration of 8.75 s from the Sylhet region.

User Query	**End-to-End Delay (s) ↓**
Query_1	5
Query_2	4
Query_3	5
Query_4	5
Query_5	4
Query_6	6
Query_7	5
Query_8	5
Query_9	5
Query_10	5
Average	**4.9**

Table 8: End-to-end delay analysis of the BanglaTalk system. Delay is calculated from the end of the user query to the start time of getting the audio response from the speech assistant.

G End-to-End delay of BanglaTalk System

To quantify the end-to-end delay of the BanglaTalk system, the delay for ten user queries is measured. Table 8 shows the results of this experiment. The BanglaTalk system has a low end-to-end delay of 4.9 s.

H User Study

To evaluate the user experience with the BanglaTalk system, a rigorous qualitative analysis is performed. Table 9 presents a summary of the experimental results, while Fig. 10 illustrates user interactions with the BanglaTalk system from two regions of Bangladesh – Sylhet and Mymensingh.

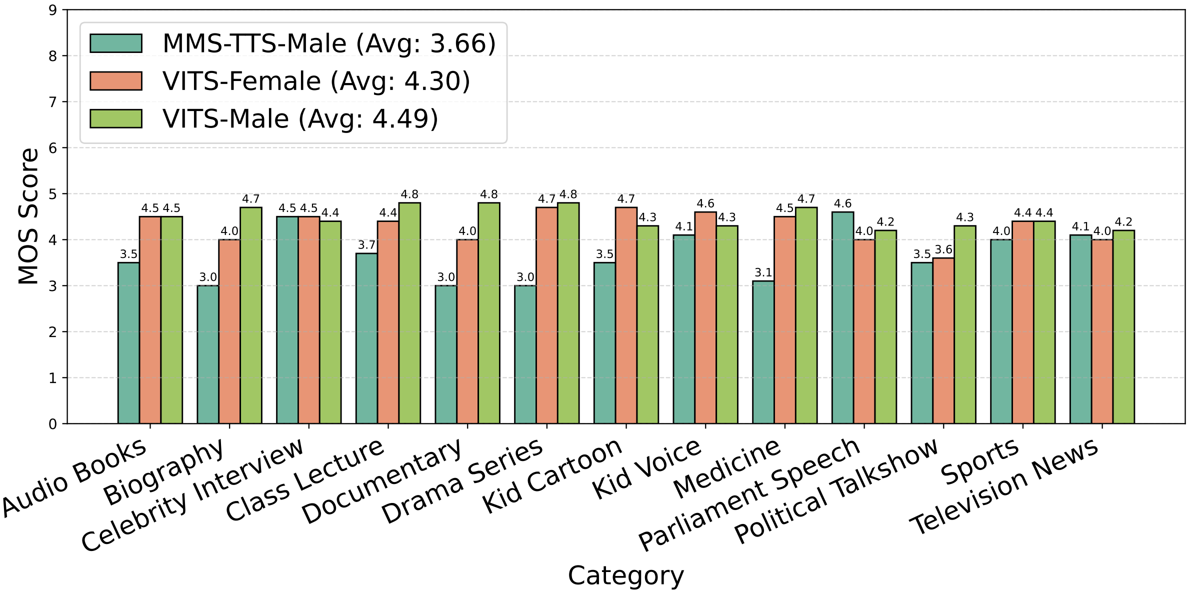
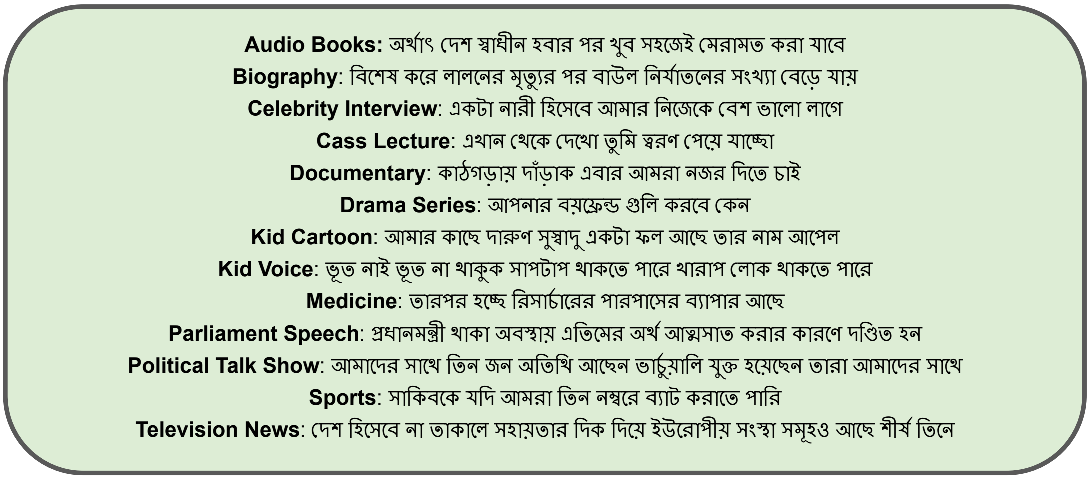


Figure 8: Mean Opinion Score (MOS) of three TTS models on the subset of the BanSpeech dataset.

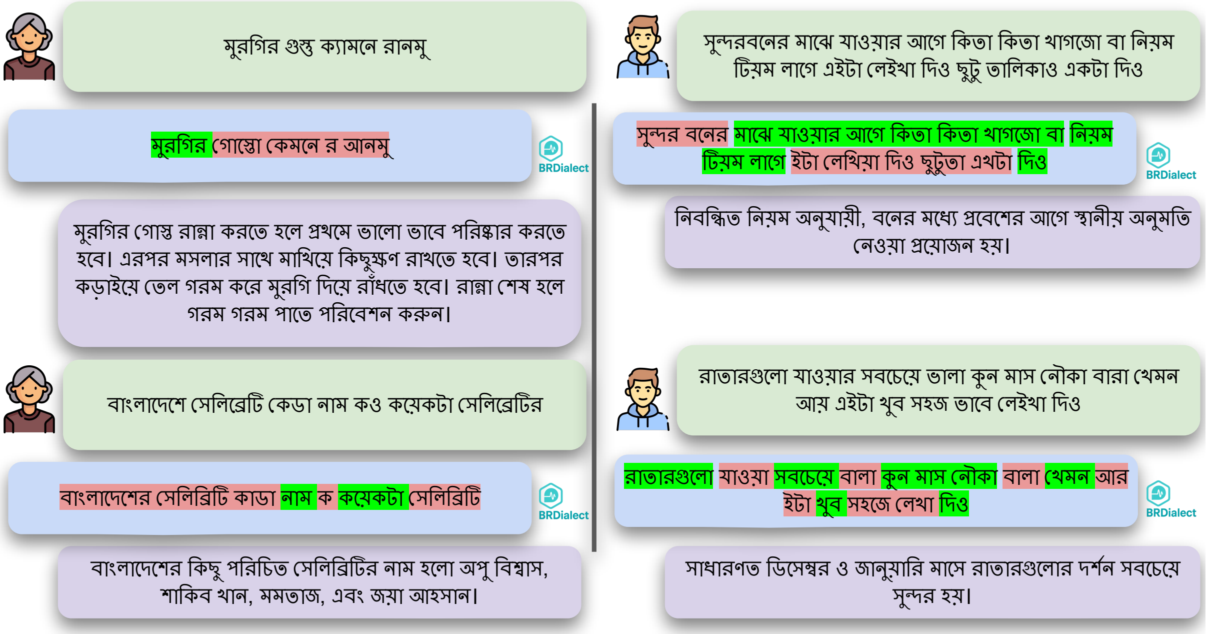


Audio Books: অর্থাৎ দেশ স্বাধীন হবার পর খুব সহজেই মেরামত করা যাবে
Biography: বিশেষ করে লালনের মৃত্যুর পর বাউল নির্যাতনের সংখ্যা বেড়ে যায়
Celebrity Interview: একটা নারী হিসেবে আমার নিজেকে বেশ ভালো লাগে
Cass Lecture: এখান থেকে দেখো তুমি ত্বরণ পেয়ে যাচ্ছো
Documentary: কাঠগড়ায় দাঁড়াক এবার আমরা নজর দিতে চাই
Drama Series: আপনার বয়ফ্রেন্ড গুলি করবে কেন
Kid Cartoon: আমার কাছে দারুণ সুস্বাদু একটা ফল আছে তার নাম আপেল
Kid Voice: ভূত নাই ভূত না থাকুক সাপটাপ থাকতে পারে খারাপ লোক থাকতে পারে
Medicine: তারপর হচ্ছে রিসার্চারের পারপাসের ব্যাপার আছে
Parliament Speech: প্রধানমন্ত্রী থাকা অবস্থায় এতিমের অর্থ আত্মসাত করার কারণে দণ্ডিত হন
Political Talk Show: আমাদের সাথে তিন জন অতিথি আছেন ভার্চুয়ালি যুক্ত হয়েছেন তারা আমাদের সাথে
Sports: সাকিবকে যদি আমরা তিন নম্বরে ব্যাট করাতে পারি
Television News: দেশ হিসেবে না তাকালে সহায়তার দিক দিয়ে ইউরোপীয় সংস্থা সমূহও আছে শীর্ষ তিনে

Figure 9: Selected text samples from the BanSpeech dataset to evaluate the performance of the TTS systems by calculating MOS scores.

User	**Region**	**Rating**					**Mean ↑**
		Query_1	Query_2	Query_3	Query_4	Query_5	
User_1	Sylhet	3.0	4.0	2.5	3.0	5	3.5
User_3	Sylhet	3.4	3.7	3.2	3.3	4.0	3.52
User_2	Mymensingh	2.0	4.0	3.8	4.2	3.3	3.46
User_4	Mymensingh	5.0	3.0	4.5	3.5	4.0	4.0
Mean							**3.62**

Table 9: User ratings on a 1–5 scale (where 1 indicates a poor experience and 5 indicates an excellent experience) for conversational interactions with the BanglaTalk speech assistant. Ratings are collected from four users representing two regional dialects – Sylhet and Mymensingh.



মুরগির গুস্ত ক্যামনে রানমু

মুরগির গোস্তো কেমনে র আনমু

মুরগির গোস্ত রান্না করতে হলে প্রথমে ভালো ভাবে পরিষ্কার করতে হবে। এরপর মসলার সাথে মাখিয়ে কিছুক্ষণ রাখতে হবে। তারপর কড়াইয়ে তেল গরম করে মুরগি দিয়ে রাঁধতে হবে। রান্না শেষ হলে গরম গরম পাতে পরিবেশন করুন।

বাংলাদেশে সেলিব্রেটি কেডা নাম কও কয়েকটা সেলিব্রেটির

বাংলাদেশের সেলিব্রিটি কাডা নাম ক কয়েকটা সেলিব্রিটি

বাংলাদেশের কিছু পরিচিত সেলিব্রিটির নাম হলো অপু বিশ্বাস, শাকিব খান, মমতাজ, এবং জয়া আহসান।

সুন্দরবনের মাঝে যাওয়ার আগে কিতা কিতা খাগজো বা নিয়ম টিয়ম লাগে এইটা লেইখা দিও ছুটু তালিকাও একটা দিও

সুন্দর বনের মাঝে যাওয়ার আগে কিতা কিতা খাগজো বা নিয়ম টিয়ম লাগে ইটা লেখিয়া দিও ছুটুতা এখটা দিও

নিবন্ধিত নিয়ম অনুযায়ী, বনের মধ্যে প্রবেশের আগে স্থানীয় অনুমতি নেওয়া প্রয়োজন হয়।

রাতারগুলো যাওয়ার সবচেয়ে ভালা কুন মাস নৌকা বারা খেমন আয় এইটা খুব সহজ ভাবে লেইখা দিও

রাতারগুলো যাওয়া সবচেয়ে বালা কুন মাস নৌকা বালা খেমন আর ইটা খুব সহজে লেখা দিও

সাধারণত ডিসেম্বর ও জানুয়ারি মাসে রাতারগুলোর দর্শন সবচেয়ে সুন্দর হয়।

Figure 10: Conversational interactions with the BanglaTalk speech assistant using the regional dialects of Mymensingh (left) and Sylhet (right). Each example includes the user's query, the transcript generated by the BRDialect ASR system, and the corresponding LLM-generated response. **Although the WER of BRDialect is relatively high in some cases, the LLM effectively captures the context and produces appropriate responses.**

Read Between the Lines: A Benchmark for Uncovering Political Bias in Bangla News Articles

Nusrat Jahan Lia
University of Dhaka
bsse1306@iit.du.ac.bd

Shubhashis Roy Dipta
University of Maryland,
Baltimore County
sroydip1@umbc.edu

Abdullah Khan Zehady
Cisco Systems
azehady@cisco.com

Naymul Islam
BanglaLLM
naymul504@gmail.com

Madhusodan Chakraborty
Maharishi International University
opuchakraborty@gmail.com

Abdullah Al Wasif
Unityflow AI
wasif@unityflow.ai

Abstract

Detecting media bias is crucial, specifically in the South Asian region. Despite this, annotated datasets and computational studies for Bangla political bias research remain scarce. Crucially because, political stance detection in Bangla news requires understanding of linguistic cues, cultural context, subtle biases, rhetorical strategies, code-switching, implicit sentiment, and socio-political background. To address this, we introduce BanglaBias, the first benchmark dataset of 200 politically significant and highly debated Bangla news articles, labeled for government-leaning, government-critique, and neutral stances, alongside diagnostic analyses for evaluating large language models (LLMs). Our comprehensive evaluation of 28 proprietary and open-source LLMs shows strong performance in detecting government-critique content (F1 up to 0.83) but substantial difficulty with neutral articles (F1 as low as 0.00). Models also tend to over-predict government-leaning stances, often misinterpreting ambiguous narratives. BanglaBias and its associated diagnostics provide a foundation for advancing stance detection in Bangla media research and offer insights for improving LLM performance in low-resource languages.[1]

1 Introduction

News media often reflect subtle political framing and bias, shaping public opinion and behavior (Fuhat and Wahab, 2024). While extensive research has addressed this problem in English by creating rich datasets for sentiment analysis (Elbouanani et al., 2025; Abercrombie and Batista-Navarro, 2020), stance detection (Rostami et al., 2025; Khiabani and Zubiaga, 2025), hate speech

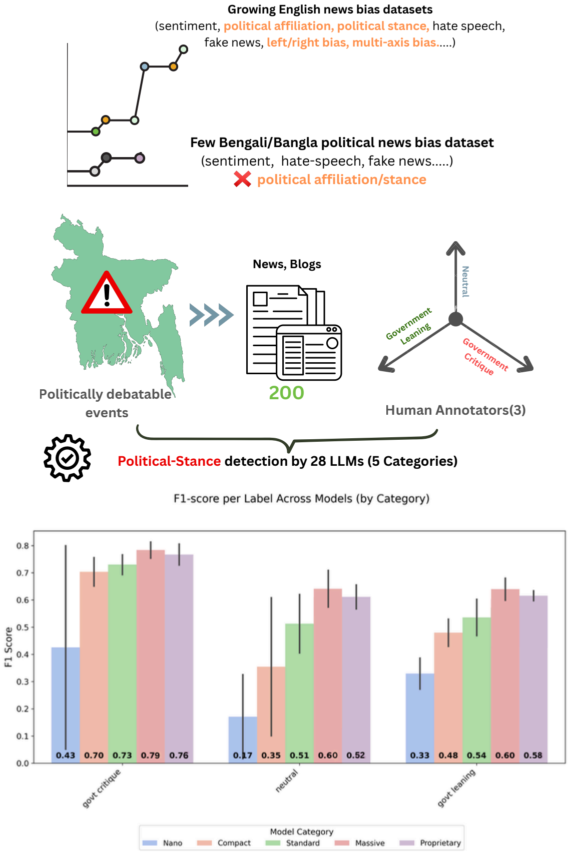


Figure 1: Overview of political stance detection study (**Growing Resources for English vs. Lack of Bangla Resource Availability**). We introduce a benchmark of 200 news articles (on politically debatable events) annotated into Government Leaning, Critique, and Neutral labels. We then evaluate performance of 28 LLMs in detecting political stance in Bengali. Performance improves with model size, with Massive and Proprietary models achieving highest F1-scores, but neutral detection remains weak. Bars for Nano models and Neutral label show **noticeably larger error ranges across models, indicating unstable performance**.

[1] https://nusrat-lia.github.io/BanglaBias/

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 61–79
December 23, 2025 ©2025 Association for Computational Linguistics

identification (Davidson et al., 2017), and political bias classification (Jones, 2024), the same cannot be said for many other languages. As Fig. 1 illustrates, Bangla news media research faces a critical gap: despite clear evidence of political framing in coverage (Islam, 2016), there exists a scarcity of annotated datasets specifically designed to detect political stance and government affiliation bias. Our evaluations in Fig. 1 revealed that while massive or proprietary models achieve strong performance on critique stance detection, neutral stance remains challenging, and Nano models showing large error bars underscores unstable performance in low-resource scenarios.

This disparity is particularly concerning given the widespread nature of political bias in Bangla news. Existing studies (both qualitative and computational) provide clear evidence that Bangla news content exhibits partisan frames that favor ruling-party narratives (Islam, 2016; Al-Zaman and Noman, 2024). For instance, two news outlets covering the same political event, such as a government policy announcement, may frame it as either a progressive reform or an authoritarian overreach, despite reporting the same factual details (Vallejo et al., 2023). Such framing differences are not merely stylistic; they represent systematic *ideological bias* – a partisan slant away from neutral reporting (McQuail and Deuze, 2020).

Computational studies have begun to tackle this with machine learning, e.g., Bangla BERT (Bhattacharjee et al., 2021) for hyper-partisanship (Mehadi Hasan et al., 2025) and innovative LLM credibility bias (Prama and Islam, 2025). However, these efforts are hampered by the lack of comprehensive and well-annotated datasets that can evaluate robust detection of political positions, specifically for the Bangladeshi socio-political landscape. Audience studies further highlight how perceived slant and censorship are reshaping Bangla news consumption (Islam et al., 2025).

To address this gap, we begin with collecting politically debatable events sourced from diverse news outlets and blogs. We then curated a dataset of 200 samples annotated by three native speakers for government stance (Government Leaning, Government Critique, or Neutral). BanglaBias addresses the challenges of detecting political stance in Bangla news, as this is far more challenging than in English due to limited prior corpora, nuanced political language, frequent code-switching with English terms, and context-dependent rhetorical styles that make bias subtle and culturally embedded. We then evaluate reasoning-based political stance detection across 28 large language models (proprietary and open-source) to assess their effectiveness for this task.

To summarize, our work revolves around three key contributions:

- We introduce the first-ever benchmark dataset for political stance detection in Bangla with comprehensive metadata and a leaderboard. The dataset provides contextual information to study how political narratives are constructed in Bangladeshi media and enables the development and evaluation of stance detection models in such a low-resource setting.
- We demonstrate multiple LLMs' effectiveness in detecting political bias in low-resource settings, with larger models achieving strong performance on Government Critique (F1=0.78) and Neutral (F1=0.61) classifications.
- We provide a systematic analysis that reveals important performance differences between different sizes of models as well as their biases and error tendencies.

2 Relevant Work

With the growing body of research on framing in NLP (Card et al., 2015; Fan et al., 2019; Baly et al., 2020; Ziems and Yang, 2021), and LLM prompting strategies (Brown et al., 2020; Achiam et al., 2023; Touvron et al., 2023) to detect stances, recent studies have explored media bias through multiple computational approaches, as we discuss in the following sections.

2.1 Framing & Political Bias in News Media

Entman (2007) defines framing as the process of shaping narratives to promote specific interpretations, which then primes audiences to think in particular ways. Computational NLP work has built on these ideas by annotating and classifying news frames and biases (Card et al., 2015; Guida et al., 2025). Early efforts include the Media Frames Corpus (Card et al., 2015) and its multilingual extension (Piskorski et al., 2023), while others focus on ideological bias, such as BASIL (Fan et al., 2019), AllSides (Baly et al., 2020), and smaller resources on regional or issue-specific perspectives (Lin et al., 2006; Chen et al., 2018; Ziems and Yang, 2021).

Such datasets do not capture the socio-political nuances of Bangladeshi cultural context, and there exists no such Bengali counterpart yet.

In practice, most NLP work treats media bias as a classification problem: one common approach is single-label classification of an article's ideology. For instance, Recasens et al. (2013) derived word-level ideological features; Spinde et al. (2021) identified biased words via embedding distances between left- and right-leaning. Transformer-based models (e.g., BERT variants) now dominate framing tasks (Liu et al., 2019; Akyürek et al., 2020; Piskorski et al., 2023). Mehadi Hasan et al. (2025) presented a Bangla BERT model fine-tuned to identify hyperpartisan (extremely biased) news articles, achieving 95.7% accuracy, using a semi-supervised approach. The authors emphasized Bangla's "low-resource" status – few annotated corpora, lexicons or pretrained models, and therefore relied on clever workarounds, i.e., machine translation (MT) for data augmentation. While MT is a practical approach, it fails to identify the socio-political facts that are specific to the region.

2.2 LLMs for Political & Subjective Analysis

Large pretrained language models have become popular for political stance and bias tasks via prompt or instruction-based methods. Researchers have evaluated models like GPT-3/GPT-4 and open alternatives (Falcon, LLaMA, Mistral, etc.) on stance detection and bias classification (Faulborn et al., 2025; Ng et al., 2025). For example, Ng et al. (2025) tested GPT-3.5 and seven open-source LLMs on three stance datasets (SemEval-2016 tweets (Mohammad et al., 2016), Elections-2016 (Sobhani et al., 2017) tweets, and the BASIL (Fan et al., 2019) news articles); they tried different prompting schemes (task description, context, chain-of-thought) and found reasoning based prompting scheme demonstrating the best performance in stance classification.

Sucu et al. (2025) showed that adding user-context information to prompts can boost performance: in a political forum stance task, contextual prompting improved accuracy by 17.5% to 38.5% over baseline. Lihammer (2023) showed GPT-3.5 and GPT-4 being able to reproduce political viewpoints. When tested in Bangladeshi context, most LLMs favored the left-leaning sources, giving higher credibility ratings (Prama and Islam, 2025). Instruction-tuned LMs can perform stance/bias tasks in a zero-shot or few-shot manner, but their outputs must be carefully validated as they may favor certain viewpoints and result in AI-induced bias in evaluating news sources, more significantly in low-resource languages like Bangla (Ng et al., 2025; Prama and Islam, 2025).

3 Creating BanglaBias

Prior research on Bangla social discourse classification (Haider et al., 2024) highlighted the challenges of transliteration noise, subtle category distinctions, and embedding reliability. Beyond these technical difficulties, a major bottleneck remains – the scarcity of high-quality, publicly available, politically nuanced Bangla datasets, which limits the development and evaluation of models for political stance detection and bias analysis. While translation or synthetic dataset generation has shown promise in domains like mathematics (Toshniwal et al., 2024), instruction following (Kim and Park, 2024), or general language tasks (Liu et al., 2024), these approaches are often ineffective for political stance detection and media bias analysis in Bangla as socio-political contexts are deeply tied to cultural, historical, and linguistic nuances that are lost or distorted in translation. Building on the research gap, we developed a systematic pipeline to create the first annotated Bangla political stance detection benchmark dataset.

For collecting the articles and human annotations, we followed these steps:

Event Selection and Categorization: We identified 46 socio-political events (full list in App. D) that generated significant public controversy or divergent viewpoints across the political spectrum. Our selection criteria prioritized events that: (1) received substantial media coverage across multiple outlets, (2) elicited clear government support or criticism in public discourse, and (3) represented diverse policy domains to ensure broad coverage of political stance patterns.

The temporal distribution spans Bangladesh's political landscape during the years 2013–2025. The early years (2013-2015) witnessed industrial disasters, the Shahbagh and Hefazat movements, and the contested 2014 election. The middle phase (2016-2019) was marked by the Holey Artisan attack, the Rohingya refugee influx, student protests on quota reform and road safety, and the 2018 election. The pandemic years (2020-2021) brought COVID-19, garment wage unrest, anti-rape protests, and growing dissent. The recent phase (2022-2025) includes fuel price hikes, the Padma Bridge opening, Rampal power plant launches, BNP–Awami League mega rallies, the boycotted 12th election, and violent quota protests with internet shutdowns and curfews.

Crawling: We implemented a crawler to collect news articles for each event title based on the year of occurrence and saved results in HTML format. If no relevant articles were retrieved for the target year, the search was extended to the following 5-6 years to capture delayed or retrospective coverage.

Parsing and normalization: The HTML files were parsed and normalized to retain only the relevant data points, excluding advertisements, navigation menus, and unrelated web content.

Article selection: For each event, at least one and up to five representative articles were selected. We prioritized articles from established mainstream outlets including Prothom Alo, The Daily Star (Bangla), Jugantor, and Kaler Kantho, alongside politically diverse sources spanning the ideological spectrum.

Human Annotation: News items were annotated for political stance using a three-category framework: *Government Leaning* (content that supports or favors government positions), *Government Critique* (content that criticizes government actions), and *Neutral* (balanced reporting without clear partisan positioning). We focused on identifying implicit stance indicators beyond explicit political statements, including source selection, framing choices, and political landscape analysis of each article's timeline and contextual emphasis.

Two annotators performed the initial annotation in parallel. Disagreements between these two annotators were resolved by a third annotator (Annotator 3), who acted as the adjudicator; the adjudicated labels are reported as final label (see annotator information in App. F).

Annotator 1 and Annotator 2 achieved 73.5% agreement (Cohen's κ = 0.574), indicating moderate inter-rater reliability. Most of the disagreements arose from neutral vs government critique and some neutral vs government leaning confusions. Disagreements were resolved by a third annotator (adjudicator) after mutual discussion and reanalysis of the political affiliations of the respective news media. Annotators primarily adhered to the stance detection decision flow outlined in App. H.

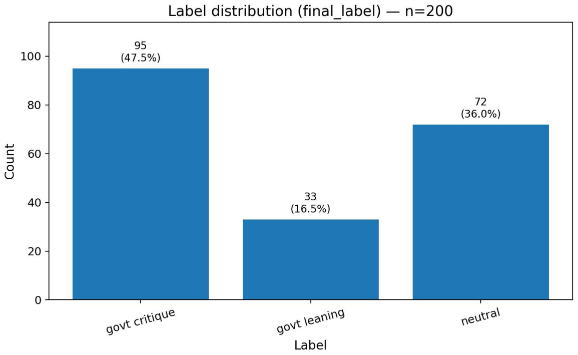


Figure 2: Distribution of three classes across the dataset: 95 Govt. Critique (47.5%), 72 Neutral (36.0%), and 33 Govt. Leaning (16.5%).

Our event-driven design captures politically significant moments where public opinion and media polarization are most visible, considering nuanced stance signals across politically diverse outlets. This also ensures quality news dataset while decreasing the chances of noisy data. **Our news collection pipeline is reusable and extensible to any low- or high-resource language**. This allows future researchers to expand the dataset with new events or apply it to other political contexts.

3.1 Dataset Statistics

In total there are 200 data samples and 8 key information; every item has a unique *ID*, a *News Body*, a *Headline*, a *Source link* and the *Date*. The *Event* field contains 46 distinct events, and *News Corpora Name* lists 54 distinct sources with the largest contributors being BBC Bangla (N=37), Prothom Alo (N=20), Jugantor (N=17), DW (N=11) and Bangla Tribune (N=11) (full details reported in Table 4). As Fig. 2 presents, the final label has three classes (govt. critique: 95, Neutral: 72, govt. leaning: 33). Because nearly half the dataset is labeled "govt. critique" while "govt. leaning" is underrepresented (≈16.5%), we recommend treating class imbalance explicitly during evaluation.

4 Experimental Setup

We benchmark the performance of several state-of-the-art large language models (LLMs) on the political stance detection using our benchmark dataset of 200 Bangla articles annotated with one of three stances: Government Leaning, Government Critique, or Neutral.

		Govt. Critique			*Neutral*			*Govt. Leaning*			*Weighted Avg.*		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
Nano	Qwen3-1.7B	0.71	0.69	0.70	0.54	0.21	0.30	0.28	0.67	0.39	0.58	0.52	0.51
	Qwen3-0.6B	0.58	0.57	0.57	0.35	0.15	0.21	0.22	0.52	0.31	0.44	0.41	0.40
Compact	TigerLLM-9B	0.74	0.78	0.76	0.62	0.65	0.64	0.62	0.45	0.53	0.68	0.68	0.68
	Qwen3-8B	0.75	0.73	0.74	0.62	0.42	0.50	0.42	0.76	0.54	0.65	0.62	0.62
Standard	Qwen3-32B	0.82	0.68	0.75	0.55	0.72	0.63	0.70	0.58	0.63	0.71	0.68	0.68
	GPT-OSS-20B	0.75	0.72	0.73	0.55	0.60	0.57	0.55	0.52	0.53	0.64	0.64	0.64
Massive	Qwen3-235B-I	0.78	0.86	0.82	0.71	0.62	0.67	0.66	0.64	0.65	0.74	0.74	0.74
	Llama3.3-70B	0.77	0.91	0.83	0.80	0.50	0.62	0.57	0.76	0.65	0.75	0.73	0.73
Proprietary	Claude-Sonnet-4	0.90	0.73	0.80	0.63	0.69	0.66	0.52	0.70	0.60	0.74	0.71	0.72
	Gemini-2.5-Pro	0.79	0.77	0.78	0.61	0.61	0.61	0.56	0.61	0.58	0.69	0.69	0.69

Table 1: Performance metrics (Precision, Recall, F1) for the top two models per category, ranked by weighted F1.

4.1 Evaluation Metrics

Our primary evaluation frames political stance detection as a Ternary Classification task. Each data point consists of an article text and a ground-truth label (Government Leaning, Neutral, or Government Critique). Given the article, models must predict the stance. We evaluate models on this task using standard metrics including accuracy, precision, recall, and weighted F1 score.

We conduct evaluations in a reasoning-based detection setting. Following Ng et al. (2025), we prompted models to classify the article and provide a brief reasoning. This framework supports additional qualitative analyses to reveal performance differences across model sizes:

(1) **Per-Label Performance** assesses F1 scores for each stance category to identify imbalances in handling Government Leaning, Neutral, or Government Critique articles.

(2) **Confusion Analysis** uses confusion matrices (aggregated by model category) to highlight common misclassification patterns, such as confusing Neutral with Government Leaning.

(3) **Bias Tendency** evaluates prediction distributions via radar plots, comparing them to the ground-truth distribution to detect systematic biases toward certain stances.

(4) **Misclassification Analysis** examines incorrect predictions alongside model-generated reasons to uncover qualitative error patterns, such as over-reliance on specific keywords or failure to detect nuance.

4.2 Models

We evaluate a diverse set of 28 language models, categorized by parameter size: Nano (<2B parameters), Compact (2B–10B), Standard (10B–40B), Massive (>40B), and Proprietary (size undisclosed). This selection includes both open-source and closed-source models to ensure comprehensive coverage.

- **Nano**: Qwen2.5-0.5B, Qwen3-0.6B, Qwen3-1.7B.
- **Compact**: Qwen2.5-3B, Qwen3-4B, Qwen3-4B-I, Qwen3-4B-T, Qwen3-8B, Llama3.1-8B, TigerLLM-9B.
- **Standard**: Qwen2.5-14B, Qwen3-14B, Qwen3-30B-I, Qwen3-30B-T, Qwen3-32B, GPT-OSS-20B.
- **Massive**: Llama3.3-70B, GPT-OSS-120B, Qwen3-235B-I, Qwen3-235B-T, GLM-4.5, DeepSeek-V3.1, DeepSeek-R1.
- **Proprietary**: Claude-Sonnet-4, Grok-4, Gemini-2.5-Pro, GPT-4.1-Mini, GPT-5-Mini.

Models are prompted to generate both a stance label (Decision: D) and reasoning (R) (R→D). Full model ids can be found in App. E. The evaluated models include families such as Qwen (Bai et al., 2023), LLaMA (Touvron et al., 2023), GLM (Zeng et al., 2025), DeepSeek (Bi et al., 2024), GPT (Achiam et al., 2023), Claude (Caruccio et al., 2024), and Gemini (Ng et al., 2025).

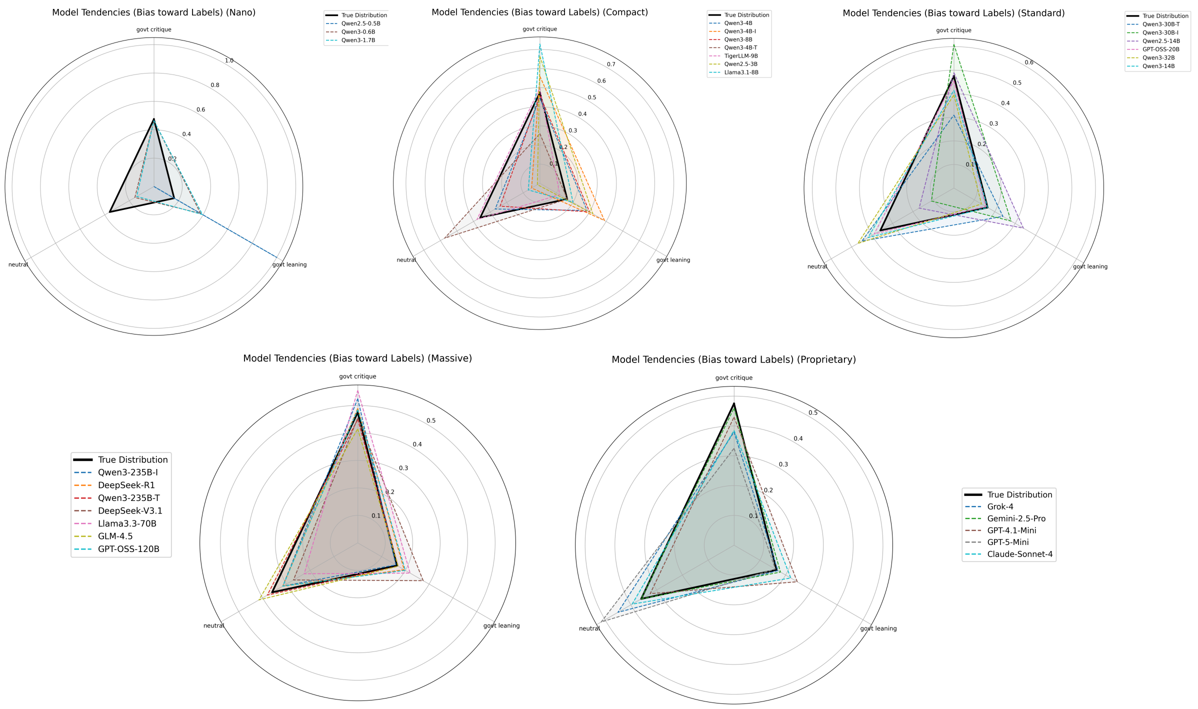


Figure 3: Radar plots showing tendencies of models (per-category) to favor particular labels, relative to the true distribution. The black polygon in each radar plot denotes the true distribution of labels and serves as the baseline.

5 Results & Analysis

5.1 Performance Metrics Analysis

Table 1 presents the performance of the top two models from each category (ranked by weighted average F1 score) in political stance detection task. Results for all 28 models can be found in Table 2.

☆ **Finding 1**: Neutral Category Exposes Model Weaknesses

Most models struggle with neutral content, with F1 scores as low as 0.00 for Qwen2.5-0.5B and 0.16 for Llama3.1-8B due to poor recall (e.g., 0.10 for Llama3.1-8B). Even massive models like Qwen3-235B-T only reach 0.68 (F1). The consistent struggle across model sizes points to potential underrepresentation of Bangla neutral samples and insufficient learning of contextual cues that distinguish neutral stances from biased ones. The challenge is particularly evident in articles requiring balanced interpretation of government actions without explicit sentiment, indicating a gap in models' ability to handle nuanced, non-polarized Bangla narratives (see Table 2).

☆ **Finding 2**: Government Critique Stance is the Easiest One to Classify

Models excel at detecting government critique stance, with top performers Llama3.3-70B and Qwen3-235B-I achieving F1 scores of 0.83 and 0.82. We hypothesize that, critical contents contain distinct linguistic cues such as strong negative sentiment and keywords (e.g., Government negligence, fraud, Repressive policy, Negligence). This strength highlights a potential bias in model design, as the same models struggle with neutral content, suggesting an over-reliance on sentiment cues rather than nuanced contextual understanding.

☆ **Finding 3**: Bias Towards Government-Leaning Stance

In government-leaning stance prediction, massive models like Qwen3-235B-I and Llama3.3-70B achieve decent F1 scores, but recall often exceeds precision (Table 2). This indicates a tendency to over-predict supportive stances. This imbalance suggests that models are overly sensitive to cues that might indicate government support, leading to false positives. This over-prediction risks am-

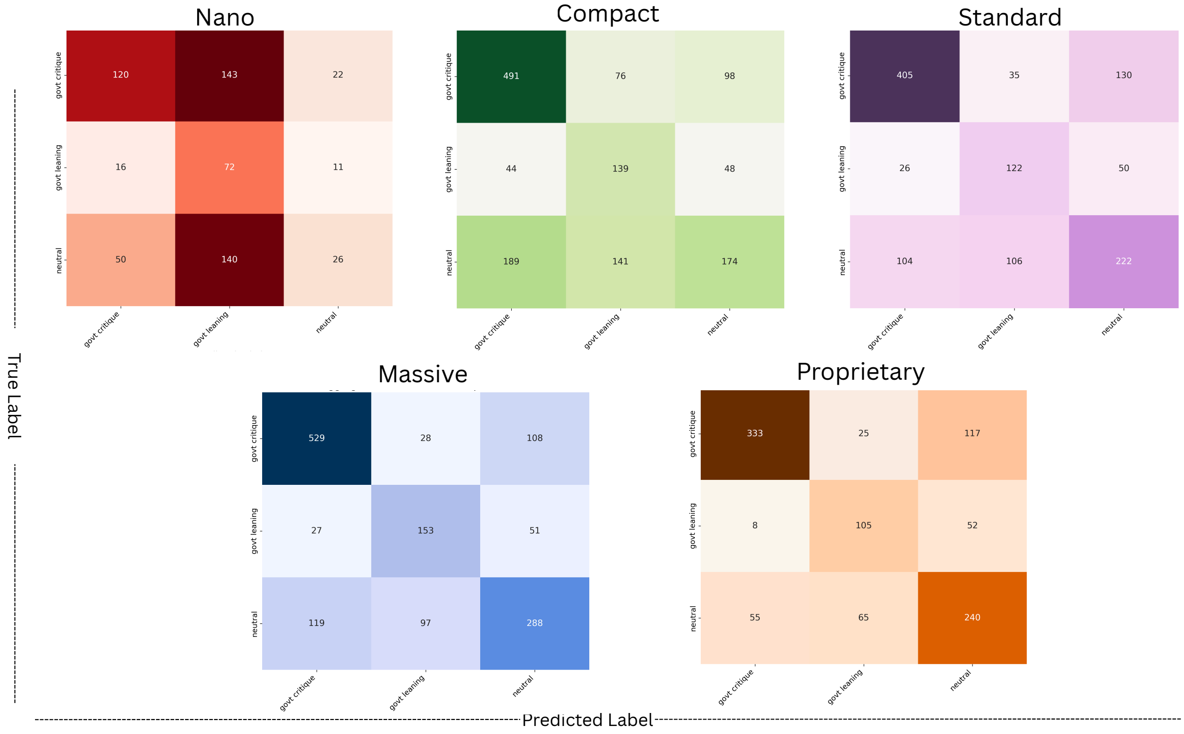


Figure 4: Aggregated Confusion Heatmap over five categories of models: Nano (3 models), Compact (7 models), Standard (6 models), Massive (7 models) and Proprietary (5 models).

plifying perceived government support in media analysis, potentially skewing automated content moderation or sentiment analysis applications.

☆ **Finding 4**: Smaller Models can be Efficient (to some extent)

Although large-scale models (Massive and Proprietary) demonstrate clear dominance in classification tasks, standard models like Qwen3-32B and even smaller TigerLLM (Raihan and Zampieri, 2025) show competitive results in detecting stances, particularly government critique and neutral. This efficiency suggests that smaller models can effectively handle key linguistic features, such as critical keywords or neutral tone indicators, without requiring the computational resources of their larger counterparts. The strong performance of TigerLLM, in particular, highlights the potential of small domain-optimized models tailored for Bangla text, which benefit from focused training on regional linguistic patterns. In low-resource settings, where computational constraints are a significant concern, such resource-efficient models can be further optimized to perform critical stance detection tasks.

5.2 Bias Tendency

Fig. 3 illustrates models' tendencies across Nano, Compact, Standard, Massive, and Proprietary categories to favor particular labels, depicting biases in understanding Bangla linguistic variations. Nano models like Qwen2.5-0.5B, Qwen3-1.7B display extreme bias toward government leaning stance. Compact models vary in biases, with TigerLLM-9B showing decent alignment. Qwen3-4B-T leans towards neutral more often, whereas Llama3.1-8B mostly mistakes neutral articles as critique stances. Standard models like Qwen2.5-14B show bias towards government leaning stance prediction, while Qwen3-30B-I struggles with over-predicting critique label. Massive models including Llama3.3-70B and Qwen3-235B-I cluster closer to the true distribution with minimal deviations. Proprietary models like Gemini-2.5-Pro and GPT-4-Mini align well overall, but GPT-5-mini leans more towards neutral label.

But Where the Bias Lies? The confusion heatmaps in Fig. 4 reveal that smaller model categories like Nano and Compact frequently confuse neutral labels with government-leaning or critique, as seen in Nano's 140 neutral instances misclassified as government-leaning and Compact's 189 neutrals mistaken for government-critique. This indicates a bias toward polarized predictions due to limited capacity. As model scale increases, such as in Standard and Massive categories, mispredictions decrease, with Massive models showing balanced performance but still confusing neutral with government-critique. Proprietary models minimize cross-polarity errors (only 25 government-critique predicted as government leaning), yet persist in confusing neutral with polar stances (55 and 65 instances).

5.3 Prompting Strategy Comparison:

To further investigate the impact of prompting on stance detection, we compare zero-shot (no example) and few-shot (multiple examples) prompting strategies across selected models, focusing on those with varying sizes (e.g., 1.7B to 70B parameters).

In subjective tasks like political bias classification in Bangla news articles, few-shot prompting can underperform compared to zero-shot, particularly in larger models (see App. I), due to risks of overfitting to example patterns, such as overemphasizing neutrality if examples are imbalanced or create conservative heuristics that dismiss implicit biases as "factual." This leads to systematic errors like defaulting to "Neutral" labels, as seen in aggregate metrics where few-shot Weighted F1 scores lag (e.g., 0.57–0.67 vs. zero-shot's 0.68–0.73). However, few-shot may still edge out in smaller models (e.g., 1.7B) needing more examples for calibration. This highlights that prompting-strategy efficacy depends on model scale, multiple patterns of framing and dataset nuances like implicit political tones.

5.4 Error Analysis

Table 3 shows some example error cases with the LLMs' reasoning. Error analysis of model predictions reveals three main failure modes.

Content-balance Ambiguity: Many misclassified articles are fact-based, presenting both government and opposition, or including technical detail that lacks an evaluative authorial stance. Such articles require understanding of external political events. For such items, the ground-truth labels often force a single polar stance, but models predict neutral as they do not detect any favoring language.

In such instances, models often reason that the article presents both parties and hence they conclude it to be neutral. Such reasoning misses the polar stance as it fails to incorporate understanding of any external entity or situational complexity.

For example, as shown in Table 3, in Article ID 110 (predicted neutral by Qwen3-32B, true label: govt. critique), the model reasoned that the article reports conflicting claims between protesters and the government-aligned student organization without taking a clear stance, presenting both accusations and denials. However, this overlooks the subtle emphasis on critiques through the framing of events, and shows how balanced presentation can mask underlying bias when external context (e.g., ongoing political tensions) is not considered.

Lexical-cue Over-reliance: When articles contain explicit praise (honorifics, laudatory framing), models predict government leaning stance even if the article was neutral. This stems from the fact that in Bangla articles, the mention of any leader name often comes with a laudatory term ("Respected Minister"), even though the articles are factual. Articles praising government initiatives when the actions were actually praise worthy, were classified as government leaning stance, despite it being neutral.

We hypothesize that as the article praises such persona, it must mean a favoring stance. Predicting such article stance is much harder as it requires world knowledge and context outside the immediate text (Burnham, 2025).

Examples include Article 98 (predicted govt. leaning by DeepSeek-V3.1, true label: neutral) (see Table 3), where the focus on a government lockdown was seen as supportive due to proactive framing, ignoring the fact based neutral trait. Similarly, in Article 148 (predicted govt. leaning by Qwen3-235B-I, true label: neutral), models reasoning highlighted government quotes against BNP, ignoring the objective situational context.

Selective Perspective Emphasis: A third common failure mode involves models amplifying one side's narrative due to disproportionate quoting or structural prominence, leading to wrong predictions in otherwise balanced articles. This occurs particularly in dynamic political contexts where opposition voices are detailed extensively, even if countered, causing models to infer critique stance.

Similarly, events framed around protests or right-abuses push models toward government critique stance prediction, despite largely factual reporting. These patterns suggest models' lack of understanding of speaker-aware representation and dependency on generic sentiment cues.

An illustrative case is Article 25 (predicted govt. critique by Qwen3-4B-T, true label: neutral) (see Table 3, where the reasoning emphasized the opposition's (BNP) detailed accusations against the government, including human rights violations, despite the news being just a factual representation of opinions. The model failed to recognize the fact based patterns, treating vivid critical language as dominant bias.

6 Conclusion

Detecting political stance in Bangla news media requires nuanced understanding of linguistic cues, cultural context, and subtle biases embedded in reporting. This paper introduces a novel benchmark dataset, BanglaBias, designed to evaluate the political stance detection capabilities of computational approaches in media bias research, which can be extended to any low-resource setting. Our findings reveal that while most LLMs excel at identifying government-critical content, they struggle significantly with neutral content due to inadequate handling of context-dependent rhetorical styles, pointing to an under-representation of extensive samples in models' training data. Our analyses indicate a tendency to misinterpret ambiguous content as government-leaning. Significant performance gains by domain-optimized LLMs further fortify our findings on the need for high-quality, culturally grounded datasets and targeted fine-tuning to improve stance detection in low-resource, politically nuanced contexts. Extending BanglaBias framework to other underrepresented languages could further democratize computational media analysis and sets the stage for more equitable, accurate, and culturally informed bias detection systems globally.

Limitations

Our study evaluates a diverse set of 28 large language models (LLMs) for political stance detection in Bangla news, covering both proprietary and open-source architectures. However, we acknowledge the rapid development of newer models or specialized architectures. We plan to extend our evaluation to incorporate emerging models and domain-specific adaptations to better capture Bangla-specific nuances.

BanglaBias, while carefully curated to represent government-leaning, government-critique, and neutral stances, may not fully capture the diversity of Bangla media, particularly from regional or less prominent outlets. Additionally, the dataset relies on articles published before the models' training cutoffs, making it challenging to ensure that models have not encountered similar content during pre-training, which could inflate performance metrics. Future work will aim to expand the dataset with more diverse sources and explore methods to verify model exposure to training data.

The task of stance detection focused on textual content, specifically article bodies, without integrating multimodal elements such as images, news-videos, or social media metadata, which often influence framing in Bangla news. We leave the exploration of multimodal stance detection to future work.

Finally, due to computational constraints, we could not perform extensive few-shot experiments, particularly proprietary ones like Claude-Sonnet-4. For consistency, we focused on models with accessible APIs or open-source weights, but future work will explore fine-tuning strategies and broader few-shot learning to improve performance, especially for neutral content detection.

References

Gavin Abercrombie and Riza Theresa Batista-Navarro. 2020. Parlvote: A corpus for sentiment analysis of political debates. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5073–5078.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Afra Feyza Akyürek, Lei Guo, Randa Elanwar, Prakash Ishwar, Margrit Betke, and Derry Tanti Wijaya. 2020. Multi-label and multilingual news framing analysis. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 8614–8624.

Md Sayeed Al-Zaman and Mridha Md Shiblee Noman. 2024. Investigating political news coverage patterns in bangladesh during 2013-2022 using computational methods. *Newspaper Research Journal*, page 07395329231221515.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Ramy Baly, Giovanni Da San Martino, James Glass, and Preslav Nakov. 2020. We can detect your bias: Predicting the political ideology of news articles. *arXiv preprint arXiv:2010.05338*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, and 1 others. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Michael Burnham. 2025. Stance detection: a practical guide to classifying political beliefs in text. *Political Science Research and Methods*, 13(3):611–628.

Dallas Card, Amber Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 438–444.

Loredana Caruccio, Stefano Cirillo, Giuseppe Polese, Giandomenico Solimando, Shanmugam Sundaramurthy, and Genoveffa Tortora. 2024. Claude 2.0 large language model: Tackling a real-world classification problem with a new iterative prompt engineering approach. *Intelligent Systems with Applications*, 21:200336.

Wei-Fan Chen, Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. 2018. Learning to flip the bias of news headlines. In *Proceedings of the 11th International conference on natural language generation*, pages 79–88.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Akram Elbouanani, Evan Dufraisse, and Adrian Popescu. 2025. Analyzing political bias in llms via target-oriented sentiment classification. *arXiv preprint arXiv:2505.19776*.

Robert M Entman. 2007. Framing bias: Media in the distribution of power. *Journal of communication*, 57(1):163–173.

Lisa Fan, Marshall White, Eva Sharma, Ruisi Su, Prafulla Kumar Choubey, Ruihong Huang, and Lu Wang. 2019. In plain sight: Media bias through the lens of factual reporting. *arXiv preprint arXiv:1909.02670*.

Mats Faulborn, Indira Sen, Max Pellert, Andreas Spitz, and David Garcia. 2025. Only a little to the left: A theory-grounded measure of political bias in large language models. *arXiv preprint arXiv:2503.16148*.

NSA Fuhat and Juliana Abdul Wahab. 2024. Mapping media framing and politics: A bibliometric analysis across the globe. *International Journal of Academic Research in Business and Social Sciences*, 14(4):1011–1032.

Matteo Guida, Yulia Otmakhova, Eduard Hovy, and Lea Frermann. 2025. Retain or reframe? a computational framework for the analysis of framing in news articles and reader comments. *arXiv preprint arXiv:2507.04612*.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, and Md Farhad Alam. 2024. Banth: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

Jahidul Islam, Hisham Anand Barbhuiya, Yeasmin Akter, Sakib Rayhan, Farhana Akter, and SM Asiful Islam Saky. 2025. Conventional media credibility and current affairs news: Emergence of alternative media in hasina-era bangladesh. *International Journal of Social Science Research & Review*, 8(4):120–135.

Sheikh Mohammad Shafiul Islam. 2016. Coverage bias of bangladesh television media: A portrayal of power and politics. *JURNAL KOMUNIKASI-MALAYSIAN JOURNAL OF COMMUNICATION*, 32(2):240–258.

Christopher Jones. 2024. Political bias dataset: A synthetic dataset for bias detection and reduction. https://huggingface.co/datasets/cajcodes/political-bias.

Parisa Jamadi Khiabani and Arkaitz Zubiaga. 2025. Cross-target stance detection: A survey of techniques, datasets, and challenges. *Expert Systems with Applications*, page 127790.

Yungi Kim and Chanjun Park. 2024. Instatrans: An instruction-aware translation framework for non-english instruction datasets. *arXiv preprint arXiv:2410.01512*.

Sebastian Lihammer. 2023. Semantic similarity comparison of political statements by chatgpt and political representatives.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander G Hauptmann. 2006. Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 109–116.

Chaoqun Liu, Wenxuan Zhang, Yiran Zhao, Anh Tuan Luu, and Lidong Bing. 2024. Is translation all you need? a study on solving multilingual tasks with large language models. *arXiv preprint arXiv:2403.10258*.

Siyi Liu, Lei Guo, Kate Mays, Margrit Betke, and Derry Tanti Wijaya. 2019. Detecting frames in news headlines and its application to analyzing news framing trends surrounding us gun violence. In *Proceedings of the 23rd conference on computational natural language learning (CoNLL)*, pages 504–514.

Denis McQuail and Mark Deuze. 2020. Mcquail's media and mass communication theory.

Mohammad Mehadi Hasan, Fatema Binte Hassan, Md Al Jubair, Zobayer Ahmed, Sazzatul Yeakin, and Md Masum Billah. 2025. Bangla bert for hyperpartisan news detection: A semi-supervised and explainable ai approach. *arXiv e-prints*, pages arXiv–2507.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 31–41.

Lynnette Hui Xian Ng, Iain J Cruickshank, and Roy Lee. 2025. Examining the influence of political bias on large language model performance in stance classification. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 19, pages 1315–1328.

Jakub Piskorski, Nicolas Stefanovitch, Giovanni Da San Martino, and Preslav Nakov. 2023. Semeval-2023 task 3: Detecting the category, the framing, and the persuasion techniques in online news in a multilingual setup. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 2343–2361.

Tabia Tanzin Prama and Md Saiful Islam. 2025. Evaluating credibility and political bias in llms for news outlets in bangladesh. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 665–677.

Nishat Raihan and Marcos Zampieri. 2025. Tigerllm-a family of bangla large language models. *arXiv preprint arXiv:2503.10995*.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, pages 1650–1659.

Peyman Rostami, Vahid Rahimzadeh, Ali Adibi, and Azadeh Shakery. 2025. Politisky24: Us political bluesky dataset with user stance labels. *arXiv preprint arXiv:2506.07606*.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 551–557.

Timo Spinde, Lada Rudnitckaia, Felix Hamborg, and Bela Gipp. 2021. Identification of biased terms in news articles by comparison of outlet-specific word embeddings. In *International Conference on Information*, pages 215–224. Springer.

Arman Engin Sucu, Yixiang Zhou, Mario A Nascimento, and Tony Mullen. 2025. Exploiting contextual information to improve stance detection in informal political discourse with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 1097–1110.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Advances in Neural Information Processing Systems*, 37:34737–34774.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Gisela Vallejo, Timothy Baldwin, and Lea Frermann. 2023. Connecting the dots in news analysis: bridging the cross-disciplinary disparities in media bias and framing. *arXiv preprint arXiv:2309.08069*.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, and 1 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*.

Caleb Ziems and Diyi Yang. 2021. To protect and to serve? analyzing entity-centric framing of police violence. *arXiv preprint arXiv:2109.05325*.

	Accuracy	*Govt. Critique*			*Neutral*			*Govt. Leaning*			*Weighted Avg.*		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
Nano Models													
Qwen2.5-0.5B	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.17	1.00	0.28	0.03	0.17	0.05
Qwen3-0.6B	0.41	0.58	0.57	0.57	0.35	0.15	0.21	0.22	0.52	0.31	0.44	0.41	0.40
Qwen3-1.7B	0.52	0.71	0.69	0.70	0.54	0.21	0.30	0.28	0.67	0.39	0.58	0.52	0.51
Compact Models													
Qwen2.5-3B	0.50	0.59	0.83	0.69	0.00	0.00	0.00	0.33	0.64	0.43	0.34	0.50	0.40
Qwen3-4B-I	0.52	0.68	0.80	0.73	0.50	0.07	0.12	0.28	0.67	0.40	0.55	0.52	0.46
Llama3.1-8B	0.55	0.58	0.88	0.70	0.50	0.10	0.16	0.45	0.55	0.49	0.53	0.55	0.47
Qwen3-4B-T	0.57	0.85	0.46	0.60	0.48	0.76	0.59	0.48	0.48	0.48	0.65	0.57	0.58
Qwen3-4B	0.58	0.73	0.68	0.71	0.56	0.42	0.48	0.39	0.67	0.49	0.61	0.58	0.59
Qwen3-8B	0.62	0.75	0.73	0.74	0.62	0.42	0.50	0.42	0.76	0.54	0.65	0.62	0.62
TigerLLM-9B	0.68	0.74	0.78	0.76	0.62	0.65	0.64	0.62	0.45	0.53	0.68	0.68	0.68
Standard Models													
Qwen2.5-14B	0.57	0.72	0.75	0.74	0.62	0.29	0.40	0.34	0.70	0.46	0.62	0.57	0.57
Qwen3-30B-I	0.62	0.70	0.89	0.78	0.77	0.24	0.36	0.41	0.70	0.52	0.68	0.62	0.59
Qwen3-14B	0.61	0.76	0.66	0.71	0.51	0.60	0.55	0.48	0.48	0.48	0.62	0.61	0.61
Qwen3-30B-T	0.61	0.85	0.56	0.68	0.51	0.64	0.57	0.50	0.73	0.59	0.67	0.61	0.62
GPT-OSS-20B	0.64	0.75	0.72	0.73	0.55	0.60	0.57	0.55	0.52	0.53	0.64	0.64	0.64
Qwen3-32B	0.68	0.82	0.68	0.75	0.55	0.72	0.63	0.70	0.58	0.63	0.71	0.68	0.68
Massive Models													
DeepSeek-V3.1	0.64	0.79	0.76	0.77	0.59	0.44	0.51	0.42	0.70	0.52	0.66	0.64	0.64
DeepSeek-R1	0.67	0.76	0.78	0.77	0.60	0.53	0.56	0.54	0.64	0.58	0.66	0.67	0.66
GPT-OSS-120B	0.67	0.76	0.78	0.77	0.59	0.51	0.55	0.55	0.67	0.60	0.66	0.67	0.66
GLM-4.5	0.69	0.82	0.72	0.76	0.60	0.69	0.65	0.59	0.61	0.60	0.70	0.69	0.69
Qwen3-235B-T	0.72	0.81	0.77	0.79	0.66	0.69	0.68	0.62	0.64	0.63	0.72	0.72	0.72
Llama3.3-70B	0.73	0.77	0.91	0.83	0.80	0.50	0.62	0.57	0.76	0.65	0.75	0.73	0.73
Qwen3-235B-I	0.74	0.78	0.86	0.82	0.71	0.62	0.67	0.66	0.64	0.65	0.74	0.74	0.74
Proprietary Models													
GPT-5-Mini	0.65	0.88	0.60	0.71	0.52	0.75	0.62	0.56	0.55	0.55	0.70	0.65	0.65
GPT-4.1-Mini	0.66	0.81	0.74	0.77	0.58	0.53	0.55	0.49	0.73	0.59	0.68	0.66	0.66
Grok-4	0.69	0.84	0.67	0.75	0.60	0.75	0.67	0.59	0.61	0.60	0.71	0.69	0.69
Gemini-2.5-Pro	0.69	0.79	0.77	0.78	0.61	0.61	0.61	0.56	0.61	0.58	0.69	0.69	0.69
Claude-Sonnet-4	0.71	0.90	0.73	0.80	0.63	0.69	0.66	0.52	0.70	0.60	0.74	0.71	0.72

Table 2: Accuracy and label-wise performance metrics of models on BanglaBias benchmark dataset. For each label (Government Critique, Neutral, Government Leaning), precision (P), recall (R), and F1-scores are reported, along with the overall weighted averages. For each block, the results are sorted based on the Weighted F1 score.

A Performance metrics of all 28 Models

Table 2 shows stance detection performance evaluation results for all 28 models (5 categories).

Article ID	Model Name	True Label	Predicted Label	Reasoning
20	Claude-Sonnet-4	Govt. Leaning	Neutral	The article provides factual reporting about Awami League's candidate announcement for the 12th national parliament election, including specific names and constituencies. It presents information objectively without editorial commentary or bias toward or against the government.
25	Qwen3-4B-T	Neutral	Govt. Critique	The article prominently features the opposition's (BNP) accusations against the government as 'unlawful, unelected, and occupying the state power' with detailed claims of human rights violations and violence. While it includes the government's response, the narrative consistently emphasizes the opposition's critique of the government's actions and policies.
98	DeepSeek-V3.1	Neutral	Govt. Leaning	The article reports the government's decision to impose a strict lockdown in a factual manner, focusing on the official announcement and the measures being taken. It presents the government as actively responding to the crisis without including significant critical perspectives or questioning the decision.
110	Qwen3-32B	Govt. Critique	Neutral	The article reports conflicting claims between protesters and the government-aligned student organization without taking a clear stance. It presents both accusations and denials objectively, focusing on factual events without evident bias.
148	Qwen3-235B-I	Neutral	Govt. Leaning	The article predominantly presents the government's perspective, portraying BNP's demands as unreasonable and disruptive, while highlighting state efforts to maintain peace and order. Government officials' statements are quoted extensively to justify decisions and frame BNP's actions as potentially inciting chaos.

Table 3: Selected cases of misclassified articles in our Bangla political stance detection study. Each row lists the article ID, the model used, the true label, the predicted label, and the reasoning provided by the model.

B Examples of Wrongly Predicted Label and Model-Reasoning

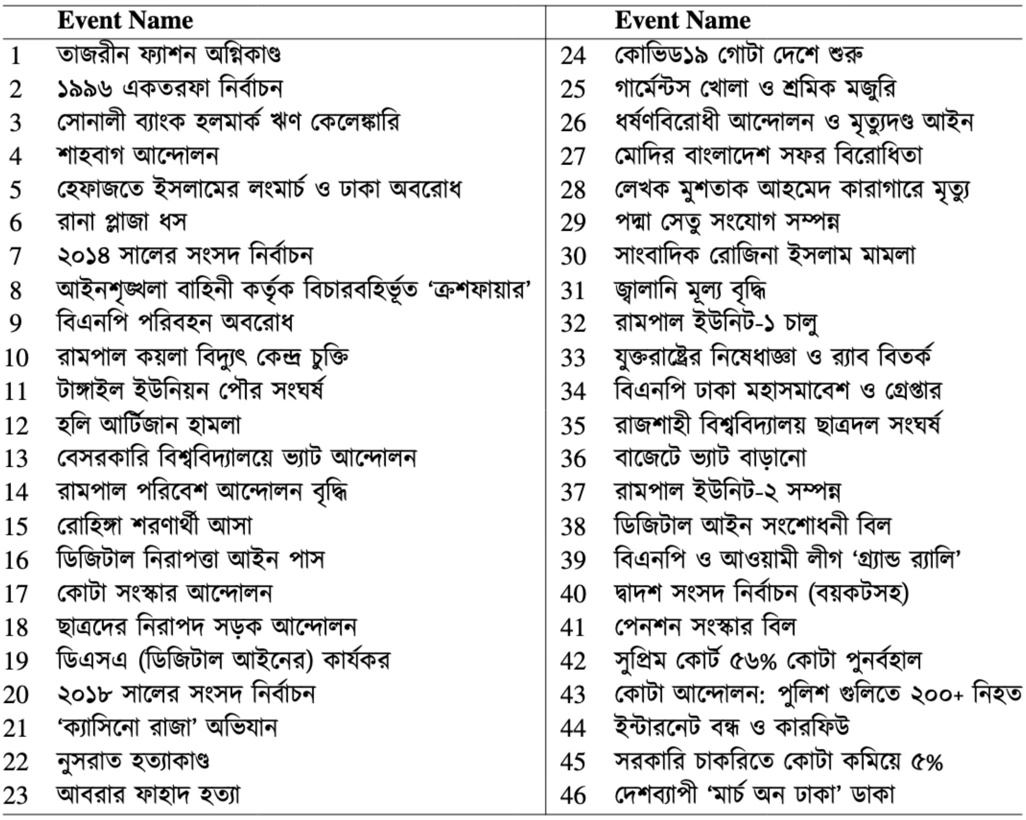
Table 3 shows five error examples along with models' reasoning.

News Source	Article Count	News Source	Article Count
BBC Bangla	37	Doinik Bangla	1
Prothom Alo	20	Bagerhatinfo	1
Jugantor	17	USEmbassy	1
DW	11	Dainik Shiksha	1
Bangla Tribune	11	Bonik Barta	1
The Daily Star	10	Bangladesh Jamate Islam	1
Samakal	7	Kaler Kontho	1
Bdnews24	7	Bdnew24	1
Dhaka Post	7	News Bangla	1
Bangla News24	7	Bangla 24 live newspaper	1
Jago News	6	Dhaka Times	1
Somoy news	5	SattAcademy	1
Dhaka Times	5	voabangla	1
Daily Ittefaq	4	Daily Janakantho	1
Daily Inqilab	4	Ajker Potrika	1
Dhaka Tribune	2	Desh Rupantor	1
Somoyer Alo	2	Daily Campus	1
Channel online	2	Khulna Gazet	1
The Daily Vorer Pata	1	Cvoice24	1
Ajkaler khobor	1	Dainik Sylhet	1
BanglaTribune	1	Dainik amader bangla	1
smsaif	1	Chalaman neywork	1
somewhereinblog	1	The doctors dialogue	1
Timetouch news	1	Kaler kontho	1
Banik Barta	1	somewhere in blog	1
DBC news	1	rajibkhaja	1
MuktiBarta	1	Green Watch BD	1

Table 4: Distribution of articles across news sources, showing the number of collected articles from each outlet.

C News Sources and Respective Article Count

Table 4 includes all the news sources and the number of articles sourced from respective corpora, blog or site.



	Event Name		Event Name
1	তাজরীন ফ্যাশন অগ্নিকাণ্ড	24	কোভিড১৯ গোটা দেশে শুরু
2	১৯৯৬ একতরফা নির্বাচন	25	গার্মেন্টস খোলা ও শ্রমিক মজুরি
3	সোনালী ব্যাংক হলমার্ক ঋণ কেলেঙ্কারি	26	ধর্ষণবিরোধী আন্দোলন ও মৃত্যুদণ্ড আইন
4	শাহবাগ আন্দোলন	27	মোদির বাংলাদেশ সফর বিরোধিতা
5	হেফাজতে ইসলামের লংমার্চ ও ঢাকা অবরোধ	28	লেখক মুশতাক আহমেদ কারাগারে মৃত্যু
6	রানা প্লাজা ধস	29	পদ্মা সেতু সংযোগ সম্পন্ন
7	২০১৪ সালের সংসদ নির্বাচন	30	সাংবাদিক রোজিনা ইসলাম মামলা
8	আইনশৃঙ্খলা বাহিনী কর্তৃক বিচারবহির্ভূত ‘ক্রশফায়ার’	31	জ্বালানি মূল্য বৃদ্ধি
9	বিএনপি পরিবহন অবরোধ	32	রামপাল ইউনিট-১ চালু
10	রামপাল কয়লা বিদ্যুৎ কেন্দ্র চুক্তি	33	যুক্তরাষ্ট্রের নিষেধাজ্ঞা ও র‍্যাব বিতর্ক
11	টাঙ্গাইল ইউনিয়ন পৌর সংঘর্ষ	34	বিএনপি ঢাকা মহাসমাবেশ ও গ্রেপ্তার
12	হলি আর্টিজান হামলা	35	রাজশাহী বিশ্ববিদ্যালয় ছাত্রদল সংঘর্ষ
13	বেসরকারি বিশ্ববিদ্যালয়ে ভ্যাট আন্দোলন	36	বাজেটে ভ্যাট বাড়ানো
14	রামপাল পরিবেশ আন্দোলন বৃদ্ধি	37	রামপাল ইউনিট-২ সম্পন্ন
15	রোহিঙ্গা শরণার্থী আসা	38	ডিজিটাল আইন সংশোধনী বিল
16	ডিজিটাল নিরাপত্তা আইন পাস	39	বিএনপি ও আওয়ামী লীগ ‘গ্র্যান্ড র‍্যালি’
17	কোটা সংস্কার আন্দোলন	40	দ্বাদশ সংসদ নির্বাচন (বয়কটসহ)
18	ছাত্রদের নিরাপদ সড়ক আন্দোলন	41	পেনশন সংস্কার বিল
19	ডিএসএ (ডিজিটাল আইনের) কার্যকর	42	সুপ্রিম কোর্ট ৫৬% কোটা পুনর্বহাল
20	২০১৮ সালের সংসদ নির্বাচন	43	কোটা আন্দোলন: পুলিশ গুলিতে ২০০+ নিহত
21	‘ক্যাসিনো রাজা’ অভিযান	44	ইন্টারনেট বন্ধ ও কারফিউ
22	নুসরাত হত্যাকাণ্ড	45	সরকারি চাকরিতে কোটা কমিয়ে ৫%
23	আবরার ফাহাদ হত্যা	46	দেশব্যাপী ‘মার্চ অন ঢাকা’ ডাকা

Figure 5: List of 46 politically debatable events (spanning diverse news coverage) included in our benchmark dataset to capture multiple perspectives.

D Politically Debated Events Covered by BanglaBias

Fig. 5 includes the list of all the events covered in BanglaBias.

Short Alias	Model ID	Category
Qwen2.5-0.5B	Qwen_Qwen2.5-0.5B-Instruct	Nano
Qwen3-0.6B	Qwen_Qwen3-0.6B	Nano
Qwen3-1.7B	Qwen_Qwen3-1.7B	Nano
Qwen2.5-3B	Qwen_Qwen2.5-3B-Instruct	Compact
Qwen3-4B	Qwen_Qwen3-4B	Compact
Qwen3-4B-I	Qwen_Qwen3-4B-Instruct-2507	Compact
Qwen3-4B-T	Qwen_Qwen3-4B-Thinking-2507	Compact
Qwen3-8B	Qwen_Qwen3-8B	Compact
Llama3.1-8B	meta-llama_Llama-3.1-8B-Instruct	Compact
TigerLLM-9B	md-nishat-008_TigerLLM-9B-it	Compact
Qwen2.5-14B	Qwen_Qwen2.5-14B-Instruct	Standard
Qwen3-14B	Qwen_Qwen3-14B	Standard
Qwen3-30B-I	Qwen_Qwen3-30B-A3B-Instruct-2507	Standard
Qwen3-30B-T	Qwen_Qwen3-30B-A3B-Thinking-2507	Standard
Qwen3-32B	Qwen_Qwen3-32B	Standard
GPT-OSS-20B	openai_gpt-oss-20b	Standard
Llama3.3-70B	meta-llama_Llama-3.3-70B-Instruct	Massive
GPT-OSS-120B	openai_gpt-oss-120b	Massive
Qwen3-235B-I	qwen_qwen3-235b-a22b-Instruct-2507	Massive
Qwen3-235B-T	qwen_qwen3-235b-a22b-Thinking-2507	Massive
GLM-4.5	z-ai_glm-4.5	Massive
DeepSeek-V3.1	deepseek_deepseek-v3.1-terminus	Massive
DeepSeek-R1	deepseek_deepseek-r1	Massive
Grok-4	x-ai_grok-4-fast	Proprietary
Claude-Sonnet-4	anthropic:claude-sonnet-4	Proprietary
Gemini-2.5-Pro	google:gemini-2.5-pro	Proprietary
GPT-4.1-Mini	openai:gpt-4.1-mini	Proprietary
GPT-5-Mini	openai:gpt-5-mini	Proprietary

Table 5: Mapping between short aliases and full model id, along with their assigned model category (Nano, Compact, Standard, Massive, or Proprietary).

E Model ID Mapping

Table 5 contains the detailed model id and categories.

F Human Annotator Details

Our annotators were graduate students from three different academic backgrounds. Each had a strong understanding of the Bangladesh's socio-political context and actively followed national media coverage. We deliberately selected annotators from diverse majors to foster inclusive knowledge exchange and ensure a multi-perspective approach to the task. None of them had any professional or institutional ties to media outlets. The annotators were fully informed about the purpose of the project and how their annotations would be used.

G Evaluation Prompt

Zero-shot evaluation prompt

```
You are a political bias classifier.

Task:
Read the following article and return ONLY
a valid JSON object with two fields:
{{
  "label": one of ["Govt leaning",
  "Neutral", "Govt critique"],
  "reason": "1–3 sentences explanation"
}}

Rules:
- Output ONLY JSON.
- Do not include markdown, code fences, or
explanations.

Article:
{{article_text}}
```

Few-shot evaluation prompt

```
You are a political bias classifier.
 Task:
 Read the following article and return
 only a valid JSON object with two fields:
{{{{
"label": one of ["Govt leaning",
"Neutral", "Govt critique"],
"reason": "1–3 sentences explanation"
}}}}

Rules:
Output ONLY JSON (no markdown, no extra
text).


Base your decision strictly on tone,
stance, and framing not just factual
accuracy.


Examples:
Example 1:
Article:
...
...
Output:
{{"label":"Neutral","reason":"..."}}
Example 2:
Article:
...
...
Output:
{{"label":"Govt leaning","reason":"..."}}
Example 3:
Article:
...
...
Output:
{{"label":"Govt
critique","reason":"..."}}

Now classify the following article:
{article_text}
```

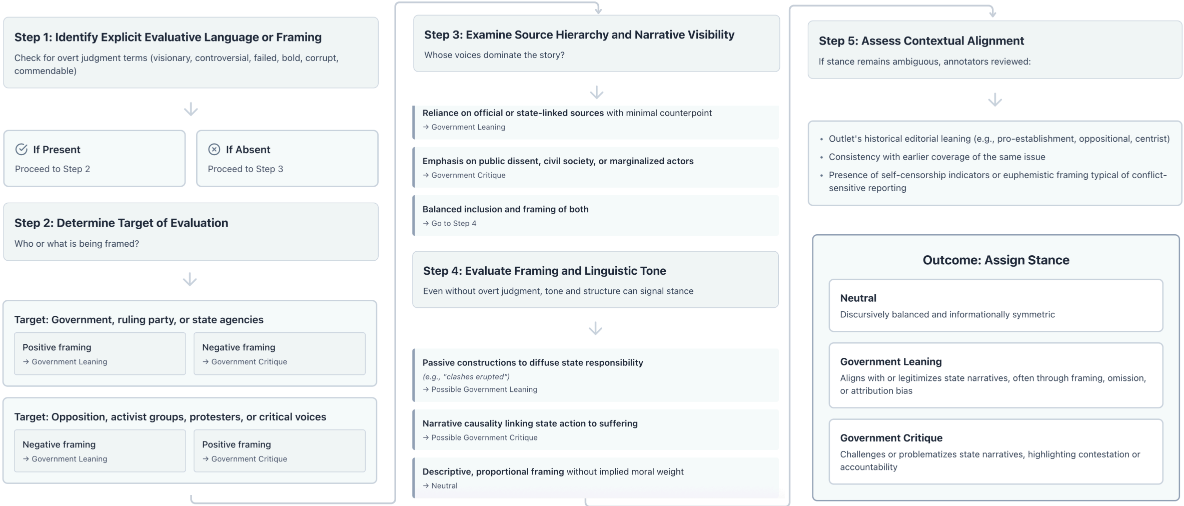

Step 1: Identify Explicit Evaluative Language or Framing
Check for overt judgment terms (visionary, controversial, failed, bold, corrupt, commendable)
If Present
Proceed to Step 2
If Absent
Proceed to Step 3
Step 2: Determine Target of Evaluation
Who or what is being framed?
Target: Government, ruling party, or state agencies
Positive framing
→ Government Leaning
Negative framing
→ Government Critique
Target: Opposition, activist groups, protesters, or critical voices
Negative framing
→ Government Leaning
Positive framing
→ Government Critique
Step 3: Examine Source Hierarchy and Narrative Visibility
Whose voices dominate the story?
Reliance on official or state-linked sources with minimal counterpoint
→ Government Leaning
Emphasis on public dissent, civil society, or marginalized actors
→ Government Critique
Balanced inclusion and framing of both
→ Go to Step 4
Step 4: Evaluate Framing and Linguistic Tone
Even without overt judgment, tone and structure can signal stance
Passive constructions to diffuse state responsibility
(e.g., "clashes erupted")
→ Possible Government Leaning
Narrative causality linking state action to suffering
→ Possible Government Critique
Descriptive, proportional framing without implied moral weight
→ Neutral
Step 5: Assess Contextual Alignment
If stance remains ambiguous, annotators reviewed:
Outlet's historical editorial leaning (e.g., pro-establishment, oppositional, centrist)
Consistency with earlier coverage of the same issue
Presence of self-censorship indicators or euphemistic framing typical of conflict-sensitive reporting
Outcome: Assign Stance
Neutral
Discursively balanced and informationally symmetric
Government Leaning
Aligns with or legitimizes state narratives, often through framing, omission, or attribution bias
Government Critique
Challenges or problematizes state narratives, highlighting contestation or accountability

Figure 6: Annotation Decision Flowchart for Political Stance Classification in the Bangladeshi Context. The decision flowchart operationalizes stance annotation within a politically polarized media system, where neutrality is a negotiated discursive position rather than a natural default.

H Annotation Decision Flowchart

A high-level annotation decision flowchart is provided in the Fig. 6.

Model	Govt. Critique			Neutral			Govt. Leaning			Weighted Avg		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Qwen3-1.7B (Few-Shot)	0.66	0.78	0.71	0.56	0.60	0.58	0.55	0.18	0.27	0.60	0.62	0.59
Qwen3-1.7B (Zero-Shot)	0.71	0.69	0.70	0.54	0.21	0.30	0.28	0.67	0.39	0.58	0.52	0.51
TigerLLM-9B (Few-Shot)	0.86	0.58	0.69	0.51	0.82	0.63	0.57	0.36	0.44	0.69	0.63	0.63
TigerLLM-9B (Zero-Shot)	0.74	0.78	0.76	0.62	0.65	0.64	0.62	0.45	0.53	0.68	0.68	0.68
Qwen3-32B (Few-Shot)	0.84	0.45	0.59	0.45	0.76	0.57	0.54	0.45	0.49	0.65	0.57	0.57
Qwen3-32B (Zero-Shot)	0.82	0.68	0.75	0.55	0.72	0.63	0.70	0.58	0.63	0.71	0.68	0.68
Llama3.3-70B (Few-Shot)	0.86	0.68	0.76	0.59	0.57	0.58	0.49	0.82	0.61	0.70	0.67	0.67
Llama3.3-70B (Zero-Shot)	0.77	0.91	0.83	0.80	0.50	0.62	0.57	0.76	0.65	0.75	0.73	0.73

Table 6: Few-Shot vs Zero-Shot Performance Comparison

I Zero-shot vs Few-shot

Zero-shot vs few-shot prompting is provided in Table 6.

BiCap: Bangla Image Captioning Using Attention-based Encoder-Decoder Architecture

Md Aminul Kader Bulbul
University of Dhaka
Dhaka-1000, Bangladesh
mdaminulkader-2017814980@cs.du.ac.bd

Abstract

Automatic image captioning has gained significant attention at the intersection of computer vision and natural language processing, yet research in low-resource languages such as Bangla remains limited. This work introduces BiCap, an attention-based encoder–decoder framework designed for Bangla image captioning. The model leverages a pretrained ResNet-50 as the encoder to extract rich visual features and a Long Short-Term Memory (LSTM) network as the decoder to sequentially generate Bangla captions. To overcome the fixed-length bottleneck of traditional encoder–decoder architectures, we integrate Bahdanau attention, enabling the decoder to dynamically focus on salient image regions while producing each word. The model is trained and evaluated on the Chitron dataset, with extensive preprocessing including vocabulary construction, tokenization, and word embedding. Experimental results demonstrate that BiCap achieves superior performance over the existing works (Masud et al., 2025; Hossain et al., 2024; Das et al., 2023; Humaira et al., 2021), yielding higher BLEU, METEOR, ROUGE, CIDEr scores. Improved fluency in human evaluation further confirms that the model generates more contextually accurate and semantically coherent captions, although occasional challenges remain with complex scenes. Recent advances in Vision–Language Models (VLMs), such as CLIP, BLIP, Flamingo, LLaVA, and MiniGPT-4, have redefined state-of-the-art captioning performance in high-resource settings. However, these models require large multimodal corpora and extensive pretraining that are currently unavailable for Bangla. BiCap therefore offers a resource-efficient, interpretable, and practically deployable solution tailored to low-resource multimodal learning.

1 Introduction

Image captioning, the task of automatically generating natural language descriptions for images, has become a central problem at the intersection of computer vision (CV) and natural language processing (NLP). It requires extracting high-level semantic information from visual inputs and mapping it into coherent textual sequences. Early captioning approaches relied on template-based methods (Kulkarni et al., 2013) or retrieval-based systems (Ordonez et al., 2011), which lacked flexibility and generalization. With the advent of deep learning, encoder-decoder architectures combining convolutional neural networks (CNNs) and recurrent neural networks (RNNs) achieved significant success (Vinyals et al., 2015). Subsequent advancements introduced attention mechanisms (Chorowski et al., 2015; Xu et al., 2015) to dynamically focus on salient image regions, alleviating the encoder-decoder bottleneck and improving semantic alignment between images and captions.

Meanwhile, the global landscape of image captioning has shifted toward Vision–Language Models (VLMs) such as CLIP (Radford et al., 2021), BLIP/BLIP-2 (Li et al., 2022, 2023), Flamingo (Alayrac et al., 2022), and multimodal LLM frameworks like LLaVA (Liu et al., 2024) and MiniGPT-4 (Zhu et al.). These systems leverage large-scale multimodal pretraining and transformer architectures to achieve near-human captioning performance. Their reliance on massive English-centric paired corpora, however, makes them impractical for Bangla, where annotated multimodal resources are extremely limited.

Despite these advancements in English, low-resource languages such as Bangla lacks large-scale multimodal corpora and pretrained vision-language models. From an NLP perspective, Bangla is morphologically rich, exhibits relatively free word order, and contains complex inflectional and derivational structures. These properties exacerbate challenges such as out-of-vocabulary (OOV) words, limited word embeddings, and poor generalization in sequence-to-sequence tasks. While efforts like

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 80–90
December 23, 2025 ©2025 Association for Computational Linguistics

BNLIT (Rahman et al., 2019) represent initial attempts at Bangla captioning, generated sentences often suffer from repetition, limited lexical diversity, and semantic incompleteness. This highlights a critical need for models that can bridge the multimodal gap while generating fluent and contextually appropriate Bangla captions.

To address this gap, this study introduces BiCap, an attention-based encoder-decoder model designed for Bangla image captioning. BiCap is designed as a resource-efficient alternative to large transformer-based vision–language architectures, offering interpretable attention patterns and competitive captioning performance in the low-resource Bangla setting.

BiCap employs a pretrained ResNet-50 (He et al., 2016) to extract rich visual embeddings, while a Long Short-Term Memory (LSTM) decoder generates sequential captions. To mitigate the bottleneck issue in traditional encoder-decoder systems, Bahdanau additive attention (Chorowski et al., 2015) is integrated, allowing the decoder to attend to different spatial regions of the image at each time step. This design ensures stronger semantic alignment between visual content and textual descriptions, improving both adequacy and fluency in Bangla captions.

Experimental validation on the Chitron dataset (Sazzed, 2020) shows that BiCap outperforms the existing baselines in terms of BLEU, METEOR, ROUGE, CIDEr scores. Human evaluation further indicates significant improvements in fluency, relevance, and semantic adequacy, confirming the effectiveness of attention in Bangla captioning.

The contributions of this study are as follows:

- An attention-based encoder–decoder framework for Bangla image captioning using ResNet-50, Bahdanau attention, and an LSTM decoder.

- Advancement of Bangla multimodal NLP by demonstrating that attention-based architectures remain effective in low-resource scenarios.

- A thorough evaluation on the largest available Bangla captioning dataset (Chitron), along with human assessment of fluency, adequacy, and relevance.

- A reusable and extensible architecture, providing a foundation for future work involving transformer decoders, multilingual embeddings, or VLM-style pretraining.

2 Related Works

The field of image captioning has been significantly advanced by the encoder-decoder architecture, which links a visual input to a natural language description. A cornerstone of this research is the work of Vinyals et al. (Vinyals et al., 2015), who introduced a complete end-to-end system utilizing a deep Convolutional Neural Network (CNN) as an encoder to extract image features and a Long Short-Term Memory (LSTM) recurrent neural network as a decoder to generate the caption. This foundational approach was further refined by subsequent research. For instance, Rennie et al. (Rennie et al., 2017) developed Self-Critical Sequence Training (SCST), a reinforcement learning method that directly optimizes non-differentiable metrics like CIDEr and BLEU, leading to substantial performance improvements by better aligning generated captions with human evaluation. Other recent advancements include convolution-free models, such as the "Full-memory transformer for image captioning" by (Lu et al., 2023), replaces the CNN encoder with a Transformer to more effectively model global visual context.

Applying these models to low-resource languages, such as Bangla, presents a unique set of challenges, primarily stemming from the lack of large, high-quality, annotated datasets and the morphological complexity of the language itself. Automatic image captioning in low-resource languages, such as Bangla, remains underexplored due to dataset scarcity and modeling challenges; a CNN–BiLSTM hybrid encoder–decoder trained on the BNLIT dataset (Rahman et al., 2019) was the first notable work in this domain, though its generalizability across broader domains remains limited. (Humaira et al., 2021) employed hybrid CNN–RNN architectures on Flickr8k and BanglaLekha datasets, reporting improved BLEU scores, though the small dataset size constrains broader applicability. A recent study (Hossain et al., 2024) proposed an image captioning approach using EfficientNetB4 and ResNet-50 for feature extraction. Another recent study (Masud et al., 2025) introduced a human-annotated dataset and an attention-driven GRU-based end-to-end model, though scalability to larger and more diverse corpora is yet to be demonstrated.

2.1 Vision–Language Models (VLMs)

Modern vision–language models (VLMs) learn joint visual–textual representations through large-scale image–text pretraining. CLIP (Radford et al., 2021) trains on 400M paired samples using contrastive learning, while BLIP (Li et al., 2022) and BLIP-2 (Li et al., 2023) unify captioning and visual grounding via transformer encoders and Q-former modules. Flamingo (Alayrac et al., 2022) incorporates cross-attention layers between a vision encoder and a frozen large language model. LLaVA (Liu et al., 2024) aligns CLIP embeddings with Vicuna to enable multimodal dialogue, and MiniGPT-4 (Zhu et al.) links ViT-G/14 with LLaMA-based decoders using lightweight alignment layers. These architectures demonstrate that multimodal reasoning requires substantial data, carefully aligned encoders–decoders, and sophisticated cross-modal fusion.

However, their applicability to Bangla is limited. Large-scale paired datasets required by these models do not exist for Bangla. The pretraining pipelines are predominantly English-centric, which leads to weak Bangla generalization even when some multilingual text is included. Moreover, the absence of Bangla multimodal benchmarks prevents effective evaluation or pretraining at VLM scale.

2.2 Large Language Models (LLMs)

Large-language-model–based captioning frameworks extend multimodal generation by leveraging powerful text decoders. OFA (Wang et al., 2022) provides a unified architecture for captioning and visual question answering, while PaLI (Chen et al.) and PaLI-X (Chen et al., 2024) scale this approach using mixture-of-experts vision–language components. GIT (Wang et al.) adopts a GPT-style decoder trained on 800M image–text pairs, enabling fluent caption synthesis at scale. InstructBLIP (Dai et al., 2023) advances this line of work by introducing instruction-following capabilities, allowing open-ended multimodal reasoning and adaptable task formats.

These systems, however, face significant constraints for Bangla captioning. Their training requires massive multimodal corpora, which are unavailable for Bangla, and multilingual tokenization struggles with Bangla's morphology, reducing downstream accuracy. Fine-tuning LLMs at this scale is computationally out of reach for most Bangla research environments, and the ecosystem lacks any open instruction-tuned multimodal model specifically optimized for Bangla tasks.

2.3 Bangla Transformer-Based Captioning

Bornon (Muhammad Shah et al., 2022) introduced a transformer-based decoder for Bangla captioning. Despite being an early effort in Bangla captioning, Bornon has several limitations that restrict its effectiveness. It is not a true multimodal Transformer and relies on global Inception-v3 features without region-level attention, resulting in weak visual grounding. The Transformer decoder also requires large datasets, but Bangla resources are limited, leading to overfitting and unstable training. In addition, Bornon uses basic word-level tokenization that poorly handles Bangla's rich morphology, causing frequent OOV and fluency issues. Its evaluation is limited to a small dataset and lacks human assessments or strong baseline comparisons.

2.4 Positioning of our proposed BiCap Architecture

Our proposed approach - BiCap is positioned as a practical and resource-efficient captioning model tailored for Bangla's low-resource setting. In contrast to VLM and LLM-based captioners that depend on massive multimodal corpora and large transformer architectures, BiCap achieves strong performance using a lightweight design. It combines ResNet-50 for efficient visual feature extraction, Bahdanau attention for fine-grained image–text alignment, and an LSTM decoder well-suited to small datasets and Bangla's morphological complexity. This makes BiCap more interpretable, more stable during training, and more effective under limited data conditions, while still remaining extensible for future integration with multilingual transformers or VLM-style modules.

3 Methodology

The BiCap model is an end-to-end multimodal architecture designed for the task of image captioning in the Bangla language. It operates on an attention-based encoder-decoder framework, meticulously crafted to synchronize the extraction of visual features with the generation of natural language sentences. The encoder, a robust convolutional neural network (CNN), processes the input image to produce a rich, spatially-distributed feature map. Subsequently, a dynamic attention mechanism acts as a spotlight, selectively highlighting the most salient visual regions pertinent to each word being generated. The decoder, a Long Short-Term Memory (LSTM) network, leverages these attended visual features and the linguistic context of previously generated words to construct fluent and semantically accurate Bangla sentences. This unified pipeline ensures a deep fusion of visual and textual information, essential for grounded and contextually relevant caption generation. The overall architecture is illustrated in Figure 1.

3.1 Encoder

The encoder component serves as the visual backbone of the BiCap model, tasked with transforming raw pixel data from an input image into a high-level, semantic representation. For this purpose, we employ a ResNet-50 (He et al., 2016) architecture, a deeply-layered CNN pre-trained on the vast ImageNet dataset. The selection of ResNet-50 is motivated by its proven efficacy in various computer vision tasks, particularly its ability to learn hierarchical features and mitigate the vanishing gradient problem through residual connections. This pre-training allows the model to leverage a vast reservoir of learned visual knowledge, which is then fine-tuned for the specific task of image captioning.

Given an input image $I \in R^{H \times W \times 3}$, where H and W are the height and width, the encoder generates a spatial feature map F:

$$F = ResNet50(I), \quad F \in R^{H' \times W' \times D} \tag{1}$$

where $H' \times W'$ are the reduced spatial dimensions of the feature map, and $D = 2048$ represents the channel depth of the final convolutional layer's output. The resulting feature map F is then reshaped into a sequence of $n = H' \times W'$ feature vectors:

$$F = \{f_1, f_2, \dots, f_n\}, \quad f_i \in R^D \tag{2}$$

Each vector f_i corresponds to a specific region of the original image, capturing localized visual information such as objects, textures, and background cues. This sequence of feature vectors retains the spatial granularity that is crucial for the subsequent attention mechanism, thereby avoiding the loss of fine-grained details that would occur if the features were condensed into a single global vector.

In this work, the encoder weights were initialized from the ImageNet-pretrained ResNet-50 model and partially fine-tuned during training. Specifically, the top convolutional block was updated, while the lower layers were kept frozen to preserve general visual representations and prevent overfitting on the relatively small Chitron dataset. This hybrid fine-tuning strategy ensures that the model adapts to the captioning domain without compromising training stability. We also observed that fully fine-tuning all layers led to faster overfitting, whereas keeping the encoder entirely frozen reduced caption diversity. The chosen partial fine-tuning configuration provides an effective trade-off between adaptability and generalization.

It is acknowledged that BiCap relies solely on a ResNet-50 encoder, which, as a pure convolutional network, primarily captures local and hierarchical visual cues. Such representations are sometimes insufficient for modeling complex global relationships among multiple objects or spatial arrangements in highly detailed scenes—an aspect crucial for generating fully contextualized captions. However, this design reflects a deliberate trade-off between representational power and resource efficiency. In the low-resource Bangla setting, transformer-based or hybrid vision encoders require substantially larger datasets and massive multimodal corpora, which are currently unavailable. The Bahdanau attention layer in BiCap helps compensate for this limitation by dynamically highlighting the most relevant regions, thereby enriching spatial context without incurring the computational cost of vision transformers. This choice ensures training stability, faster convergence, and interpretable attention maps. In the future work, we will explore lightweight CNN–ViT hybrids or multi-scale attention mechanisms to further enhance global scene understanding once larger Bangla multimodal datasets become accessible.

3.2 Attention Mechanism

A common challenge in encoder-decoder architectures is the bottleneck problem, where compressing all visual information into a single fixed-size vector can lead to the loss of fine-grained spatial and semantic details. To circumvent this, the BiCap model integrates an attention mechanism, which allows the decoder to dynamically focus on the most relevant parts of the image at each step of the caption generation process. We adopt the Bahdanau additive attention (Chorowski et al., 2015) mechanism due to its effectiveness in aligning visual and textual modalities.

At each decoding time step t, the attention mechanism computes an alignment score $e_{t,i}$ between

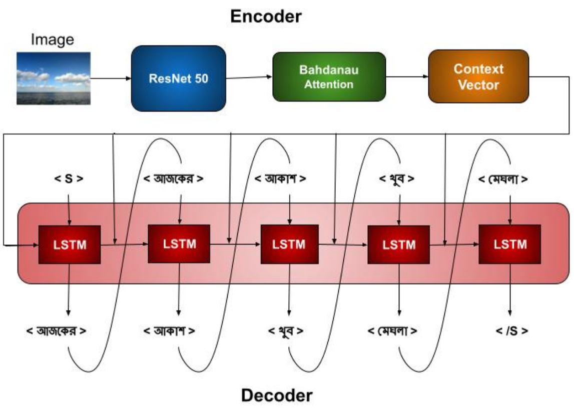


Figure 1: Overview of the BiCap architecture: ResNet-50 encoder extracts spatial features, Bahdanau attention computes context vectors, and an LSTM decoder generates captions sequentially.

the previous decoder hidden state h_{t-1} and each encoder feature vector f_i. This score quantifies the relevance of the i-th visual region to the next word to be generated. The alignment score is calculated as follows:

$$e_{t,i} = v_a^\top \tanh(W_h h_{t-1} + W_f f_i) \tag{3}$$

where $W_h \in R^{k \times d}$ and $W_f \in R^{k \times D}$ are learnable weight matrices, and $v_a \in R^k$ is a trainable vector. Here, d is the hidden dimension of the decoder and D is the dimension of the encoder features.

The alignment scores are then normalized using a softmax function to obtain a set of attention weights $\alpha_{t,i}$:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{n} \exp(e_{t,j})} \tag{4}$$

These weights, which sum to one, represent a probability distribution over the encoder's feature vectors. They indicate how much attention the decoder should pay to each visual region. The context vector c_t is then computed as a weighted sum of the encoder features, with the attention weights acting as coefficients:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} f_i \tag{5}$$

The context vector c_t is a dynamically created representation of the most salient visual information, specifically tailored for the generation of the word at time step t. This mechanism effectively simulates a human-like visual focus, ensuring that the decoder's output is not only syntactically correct but also semantically grounded in the visual content.

3.3 Decoder

The decoder's primary role is to translate the multimodal representation into a coherent and linguistically fluent Bangla sentence. For this, we utilize a Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997), a type of recurrent neural network particularly well-suited for processing sequential data. LSTMs are capable of capturing long-range dependencies, which is critical for handling the rich morphology and flexible syntactic structure of the Bangla language.

At each time step t, the decoder takes as input the concatenation of the word embedding of the previous token $E(y_{t-1})$ and the attention-derived context vector c_t:

$$x_t = [E(y_{t-1}); c_t] \tag{6}$$

where $E(\cdot)$ is a word embedding function that maps each word in the vocabulary to a dense vector space. This combined input provides the LSTM with both the linguistic context (from the previous word) and the relevant visual context (from the attention mechanism). The LSTM then updates its hidden state h_t and cell state s_t based on the current input and the previous states:

$$h_t, s_t = LSTM(x_t, h_{t-1}, s_{t-1}) \tag{7}$$

The final hidden state h_t is then used to predict the next word in the sequence. This is achieved by passing the hidden state through a fully connected layer followed by a softmax function to produce a probability distribution over the entire vocabulary:

$$p(y_t \mid y_{1:t-1}, I) = Softmax(W_o h_t + b_o) \tag{8}$$

where W_o and b_o are learnable output parameters.

During training, we employ the teacher forcing technique, where the ground-truth word y_{t-1} is fed as input at each time step. This method stabilizes the training process and accelerates convergence by providing the model with correct historical context. In contrast, during inference, the model operates autoregressively, feeding its own predicted word from the previous step as input. This process continues until an end-of-sequence token is generated, at which point the complete Bangla caption is formed. The tight synchronization between the encoder's visual understanding, the attention mechanism's dynamic focus, and the decoder's linguistic generation is what enables the BiCap model to produce high-quality, descriptive Bangla captions.

We acknowledge that the LSTM-based decoder represents a sequential, Recurrent Neural Network (RNN) architecture. Such models process inputs step by step, which can limit their ability to capture very long-range dependencies and make them slower to train than fully parallelized Transformer decoders. Moreover, this architecture does not directly exploit the large-scale multimodal representations available in modern Transformer-based vision–language models such as BLIP or CLIP.

However, this design choice is intentional and well-motivated for Bangla captioning under low-resource conditions. Transformer-based decoders typically require millions of image–text pairs and powerful GPUs to train effectively, while the available Bangla datasets (e.g., Chitron) are relatively small. In such cases, RNN-based models like LSTM remain more stable, data-efficient, and interpretable. The combination of Bahdanau attention and LSTM allows BiCap to focus on semantically relevant regions of the image while maintaining grammatical and morphological accuracy in Bangla output. Furthermore, the sequential structure enables explicit attention visualization, offering interpretability that is often opaque in Transformer architectures. As Bangla multimodal resources expand, BiCap can be adapted to integrate Transformer or hybrid decoding modules to improve scalability and contextual reasoning in its future extensions.

4 Experimental Setup

4.1 Dataset

All experiments are conducted on the **Chitron** dataset (Sazzed, 2020), the largest publicly available Bangla image captioning corpus. It consists of 15,438 images with human-annotated Bangla captions, collected from diverse sources including news portals, Wikipedia Commons, and social media. The images span a variety of domains such as people, objects, scenes, and events, making it a suitable benchmark for Bangla captioning.

For this study, the dataset is partitioned into training (80%), validation (10%), and test (10%) splits. Table 1 summarizes the dataset statistics.

	Images & Captions	Vocabulary Size	Max Caption Length
Training	12350	9800	20
Validation	1544	—	20
Test	1544	—	20

Table 1: Chitron dataset statistics and experimental split.

4.2 Preprocessing

All images were resized to 299×299 pixels before feature extraction to ensure consistent input dimensions for the ResNet-50 encoder. This resolution is a standard preprocessing step inherited from the ImageNet training configuration, which ResNet-50 expects for optimal performance. Resizing maintains a uniform aspect ratio across images, reduces computational overhead, and enables efficient batch processing without affecting semantic content. Using 299×299 therefore ensures compatibility with pretrained convolutional filters while balancing accuracy and efficiency during training.

Captions are tokenized using a rule-based Bangla tokenizer that handles whitespace, punctuation, and compound words. Rare words (frequency < 5) are replaced with an `<UNK>` token, resulting in a vocabulary size of approximately 9.8K unique tokens. Special tokens `<START>` and `<END>` are appended to mark caption boundaries, and all sequences are padded or truncated to a maximum length of 20 tokens. Here, the maximum caption length was truncated to 20 tokens to standardize sequence lengths for training and reduce computational complexity. Analysis of the Chitron dataset revealed that most Bangla captions fall below this length, so truncation affects very few instances while allowing efficient batch processing and stable LSTM training. Limiting the token size also mitigates the risk of overfitting on rare, excessively long sequences and ensures that the attention mechanism focuses on the most semantically relevant words in the caption. Future work may explore dynamic sequence lengths or subword tokenization to better accommodate longer captions without compromising efficiency.

We acknowledge that BiCap uses a simple frequency-based preprocessing strategy, where rare words (frequency < 5) are replaced with a generic token, yielding a vocabulary of approximately 9.8K words. This approach may limit the model's ability to generate captions containing rare or morphologically complex Bangla words, potentially reducing linguistic diversity. However, this design choice was made to stabilize training and prevent sparsity in the embedding layer given the small dataset size. The model's focus is on producing semantically accurate and fluent captions rather than exhaustive lexical coverage. Despite the reduced vocabulary, BiCap achieves strong fluency and adequacy scores in both automatic and human evaluations, indicating that the simplified preprocessing did not significantly harm overall descriptive quality. Future work will explore subword-based tokenization and morphological segmentation to better capture Bangla's rich morphology and enhance lexical diversity.

BiCap incorporates several Bangla-specific adaptations across preprocessing and model design to better handle the language's rich morphology and script characteristics. During preprocessing, Unicode normalization and punctuation cleaning were applied to accommodate compound and conjunct characters in Bangla text. Tokenization was performed using a custom rule-based segmenter designed for Bangla word boundaries, as existing multilingual tokenizers often split characters incorrectly. In the decoder, Bangla word embeddings were trained from scratch to capture morphological variations such as inflection and postpositions, which differ from English syntax. The attention-based decoder further aids in aligning Bangla sentence structure with image regions, improving grammatical fluency. These tailored adjustments ensure that BiCap learns robust visual–linguistic associations specific to Bangla rather than relying on multilingual defaults.

4.3 Model Training

The encoder uses a pretrained ResNet-50 truncated before the final classification layer, while the decoder is a single-layer LSTM with hidden size 512 and embedding dimension 300. The attention module employs a hidden alignment size of 256. Dropout ($p = 0.5$) is applied to embeddings and LSTM outputs. The optimizer is Adam with learning rate 5×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and batch size 64.

4.4 Implementation Details

The model is implemented in `PyTorch` 1.11 and trained on an NVIDIA Tesla P100 GPU with 16 GB memory. Each training epoch requires approximately 10 minutes, and full training converges within 30-35 epochs.

5 Results and Discussion

This section provides a rigorous, data-driven evaluation of the proposed BiCap framework. A comprehensive set of established metrics (BLEU, METEOR, ROUGE, CIDEr), coupled with human judgments, is employed to quantify performance. The results of BiCap are systematically benchmarked against several state-of-the-art baseline models (Masud et al., 2025; Hossain et al., 2024; Das et al., 2023; Humaira et al., 2021). This comparative analysis is specifically designed to isolate and highlight the significant performance gains attributable to the incorporation of the Bahdanau attention mechanism in encoder-decoder model within the context of Bangla image captioning.

We acknowledge that the model has been trained and evaluated exclusively on the Chitron dataset, which, despite being the largest resource available, remains relatively small compared with datasets used in high-resource languages. This limitation naturally raises concerns regarding generalizability to unseen or cross-domain image distributions. However, this experimental focus is deliberate: BiCap is explicitly designed and optimized for low-resource multimodal settings, where large-scale Bangla datasets are unavailable. By operating effectively within such constraints, BiCap demonstrates the feasibility of attention-based captioning under realistic data scarcity conditions. Moreover, the model's attention maps and human evaluation results indicate that it learns meaningful visual–textual correspondences rather than overfitting to dataset bias. Future work will extend evaluation to additional Bangla or multilingual datasets and explore transfer learning or synthetic data augmentation to further enhance robustness and domain generalization.

5.1 Quantitative Results

To further assess the effectiveness of BiCap, we compare its performance against several existing Bangla image captioning models: (Masud et al., 2025; Hossain et al., 2024; Das et al., 2023; Humaira et al., 2021). Evaluation is conducted using standard captioning metrics: BLEU-1 to BLEU-4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), ROUGE (Lin, 2004), and CIDEr (Vedantam et al., 2015). Table 2 summarizes the results.

BiCap achieves state-of-the-art results across all evaluation metrics. Specifically, it records a BLEU-4 score of 32.73, which is over 10 points higher than the strongest baseline (Masud et al., 2025). Improvements are also consistent in METEOR (+0.03) and ROUGE (+0.06), indicating not only higher n-gram overlap but also better recall-oriented performance and semantic adequacy. The CIDEr score of 0.29 further demonstrates that BiCap produces captions more closely aligned with human references compared to earlier works.

The gains in BLEU-1 and BLEU-2 suggest that BiCap excels at generating accurate local word choices, while improvements in BLEU-3 and BLEU-4 highlight its ability to maintain coherence over longer n-grams. This reflects the effectiveness of the Bahdanau attention mechanism, which dynamically focuses on different visual regions during decoding, thereby avoiding the information bottleneck present in earlier CNN–RNN systems.

Another factor behind BiCap's superior performance is the use of a pretrained ResNet-50 encoder, which provides rich, high-level semantic embeddings that generalize well to diverse image domains in the Chitron dataset. In contrast, earlier works often relied on shallower CNNs or training from scratch, which limited their ability to capture complex visual concepts.

Finally, the integration of attention with an LSTM decoder enables BiCap to handle the morphological richness and free word order of Bangla more effectively than prior models. This leads to higher lexical diversity, fewer incomplete sentences, and improved alignment between visual objects and their textual descriptions.

Overall, the quantitative results confirm that BiCap sets a new benchmark for Bangla image captioning, outperforming methods developed between 2021 and 2025 due to its stronger visual representations, attention-driven alignment, and language-aware decoding strategy.

5.2 Human Evaluation

To complement automatic scores, 20 randomly sampled image–caption pairs from the test set were rated by 83 native Bangla speakers on three dimensions: *fluency* (grammatical correctness), *adequacy* (coverage of salient objects/events), and *relevance* (semantic alignment with the image). Ratings were given on a 5-point Likert scale.

Table 3 summarizes the results. BiCap surpasses the baseline across all three criteria, with particularly strong gains in adequacy and relevance, confirming that attention improves semantic grounding. This clarification emphasizes that the evaluation focused on the quality of the generated captions in context of their corresponding images, not on standalone image selection. Such human ratings complement the automatic metrics and provide a more holistic measure of caption quality.

5.3 Performance Analysis

The enhanced performance of the BiCap model is attributed to three primary architectural features:

1. **Rich Feature Extraction:** The use of a pretrained ResNet-50 encoder allows BiCap to extract rich, hierarchical visual features, enabling the capture of finer details (e.g., attributes, relations) and the mention of multiple objects in complex scenes.

2. **Dynamic attention:** By attending to different image regions at each decoding step, BiCap avoids the information bottleneck of global feature vectors.

3. **Better sequence modeling:** The LSTM decoder, conditioned on context vectors, produces more coherent sequences compared to the baseline RNN.

5.4 Limitations and Future Works

Despite the aforementioned advancements, the current BiCap implementation faces several limitations that highlight clear paths for future research to further enhance its accuracy, diversity, and robustness:

1. As the encoder relies on ResNet-50, BiCap primarily captures local rather than fully global visual dependencies. Future extensions will explore hybrid CNN–ViT architectures for improved scene understanding.

Models	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	RUOUGE	CIDEr
BiCap (Proposed)	**71.42**	**59.31**	**46.40**	**32.73**	**0.37**	**0.42**	**0.29**
Masud et al., 2025	62.74	51.63	39.81	22.28	0.34	0.36	0.23
Hossain et al., 2024	59.37	47.26	31.74	20.39	0.29	0.31	0.21
Das et al., 2023	53.46	42.83	27.63	20.79	0.27	0.34	0.21
Humaira et al., 2021	51.37	41.94	26.07	19.45	0.24	0.33	0.20

Table 2: Quantitative evaluation using BLEU, METEOR, ROUGE and CIDEr scores on the Chitron dataset.

Models	Fluency	Adequacy	Relevance
BiCap (Proposed)	**4.3**	**4.4**	**4.6**
Masud et al., 2025	3.6	3.3	3.9
Hossain et al., 2024	3.7	2.8	2.9
Das et al., 2023	3.7	3.1	3.4
Humaira et al., 2021	3.4	2.6	2.4

Table 3: Human evaluation of generated captions (Likert scale 1-5).

2. While effective in low-resource settings, BiCap's RNN-based decoder is inherently sequential and may not fully capture long-range dependencies. Future work will explore hybrid Transformer–RNN extensions.

3. Although BiCap performs well on the 15K-image Chitron dataset, its generalizability to unseen domains remains unverified due to limited Bangla multimodal data. Future work will focus on additional Bangla or multilingual datasets, cross-domain evaluation and multilingual transfer learning to enhance robustness.

6 Conclusion

This study introduced BiCap, an attention-based encoder-decoder framework for Bangla image captioning. By combining ResNet-50 for visual feature extraction, Bahdanau attention for dynamic region selection, and an LSTM decoder for sequential text generation, BiCap addresses key challenges in Bangla captioning such as semantic grounding and syntactic fluency.

Experimental results on the Chitron dataset demonstrated that BiCap significantly outperforms the existing baselines, achieving higher BLEU, METEOR, ROUGE and CIDEr scores. Superior human evaluation ratings further highlighted its ability to generate more descriptive and contextually accurate Bangla captions.

Overall, BiCap establishes a strong foundation for research in low-resource multimodal learning, particularly for Bangla. Future extensions could integrate transformer-based decoders, leverage multilingual embeddings for rare word coverage, and explore larger-scale multimodal datasets to further enhance caption quality. This work represents a step forward in bridging the gap between computer vision and natural language processing for under-represented languages.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, and 1 others. 2024. On scaling up a multilingual vision and language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14432–14444.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, and 1 others. Pali: A jointly-scaled multilingual language-image model. In *The Eleventh International Conference on Learning Representations*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. *Advances in neural information processing systems*, 28.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N

Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems*, 36:49250–49267.

Bidyut Das, Ratnabali Pal, Mukta Majumder, Santanu Phadikar, and Arif Ahmed Sekh. 2023. A visual attention-based model for bengali image captioning. *SN Computer Science*, 4(2):208.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Md Anwar Hossain, Mirza AFM Rashidul Hasan, Sajeeb Kumar Ray, and Naima Islam. 2024. Generating bangla image captions with deep learning techniques. In *2024 6th International Conference on Sustainable Technologies for Industry 5.0 (STI)*, pages 1–6. IEEE.

Mayeesha Humaira, Paul Shimul, Md Abidur Rahman Khan Jim, Amit Saha Ami, and Faisal Muhammad Shah. 2021. A hybridized deep learning method for bengali image captioning. *International Journal of Advanced Computer Science and Applications*, 12(2).

Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2013. Babytalk: Understanding and generating simple image descriptions. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2891–2903.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven CH Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning (ICML)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, and 1 others. 2024. Llava-plus: Learning to use tools for creating multimodal agents. In *European conference on computer vision*, pages 126–142. Springer.

Tongwei Lu, Jiarong Wang, and Fen Min. 2023. Full-memory transformer for image captioning. *Symmetry*, 15(1):190.

Adiba Masud, Md Biplob Hosen, Md Habibullah, Mehrin Anannya, and M Shamim Kaiser. 2025. Image captioning in bengali language using visual attention. *PloS one*, 20(2):e0309364.

Faisal Muhammad Shah, Mayeesha Humaira, Md Abidur Rahman Khan Jim, Amit Saha Ami, and Shimul Paul. 2022. Bornon: Bengali image captioning with transformer-based deep learning approach. *SN Computer Science*, 3(1):90.

Vicente Ordonez, Girish Kulkarni, and Tamara Berg. 2011. Im2text: Describing images using 1 million captioned photographs. *Advances in neural information processing systems*, 24.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamila Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*.

A. Rahman and 1 others. 2019. Bnlit: Bangla natural language image-to-text dataset and baseline. In *Proceedings of the International Conference on Bangla Speech and Language Processing (ICBSLP)*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024.

Sumon Sazzed. 2020. Chittron: A bangla image captioning dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*, pages 751–758.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language. *Transactions on Machine Learning Research*.

Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International conference on machine learning*, pages 23318–23340. PMLR.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In *The Twelfth International Conference on Learning Representations*.

Zero-Shot Multi-Label Classification of Bangla Documents: Large Decoders Vs. Classic Encoders

Souvika Sarkar[1], Md. Najib Hasan[1], Santu Karmaker[2]
[1] Accessible AI Lab (A2I Lab), School of Computing, Wichita State University
[2] Bridge-AI Lab, Department of Computer Science, University of Central Florida
souvika.sarkar@wichita.edu, mxhasan39@shockers.wichita.edu, santu@ucf.edu

Abstract

Bangla, a language spoken by over 300 million native speakers and ranked as the sixth most spoken language worldwide, presents unique challenges in natural language processing (NLP) due to its complex morphological characteristics and limited resources. Although recent large-decoder-based LLMs, such as GPT, LLaMA, and DeepSeek, have demonstrated excellent performance across many NLP tasks, their effectiveness in Bangla remains largely unexplored. In this paper, we establish the first benchmark comparing large decoder-based LLMs with classic encoder-based models for the Zero-Shot Multi-Label Classification (Zero-Shot-MLC) task in Bangla. Our evaluation of 32 state-of-the-art models reveals that existing so-called powerful encoders and decoders still struggle to achieve high accuracy on the Bangla Zero-Shot-MLC task, suggesting a need for more research and resources for Bangla NLP.

1 Introduction

Bangla is the sixth most spoken language worldwide, with over 300 million native speakers, accounting for nearly 4% of the global population[1]. Despite its widespread use, Bangla remains a low-resource language in the domain of Natural Language Processing (NLP) (Joshi et al., 2020), with limited benchmark studies and scarce linguistic resources. The rise of classic encoders, such as LASER (Artetxe and Schwenk, 2019) and LaBSE (Feng et al., 2020), alongside large decoder-based models such as GPT (Brown et al., 2020), BLOOM (Scao et al., 2022), LLaMA (Touvron et al., 2023), and DeepSeek (DeepSeek-AI et al., 2025), has driven significant advancement in multilingual NLP. However, while these models have demonstrated exceptional performance in high-resource languages such as English, Chinese, and Spanish, their effectiveness in low-resource languages, like Bangla, remains largely unexplored. This gap raises a crucial question: How well do LLMs perform on Bangla-specific tasks compared to classic encoder-based models?

To address this, we present a benchmarking study, evaluating 32 state-of-the-art models on the Zero-Shot Multi-Label Classification (Zero-Shot-MLC) task for Bangla, including three classic encoders—LaBSE (Feng et al., 2020), LASER (Artetxe and Schwenk, 2019), and BanglaTransformers (Muhammad Rafsan Kabir et al., 2024)—and 29 large decoder-based LLMs. Our findings reveal that while a few models achieve F_1 scores around 60%, most struggle with reliable classification, highlighting significant gaps in multilingual adaptability. This study provides the first systematic decoder vs. encoder evaluation for Bangla Zero-Shot-MLC, emphasizing the need for further research and resources in Bangla NLP. Our dataset and source code are publicly available at: https://github.com/MdNajibHasan/BanglaNLP.git.

2 Background and Related Works

Bangla NLP and Zero-shot classification: In the domain of Bangla NLP, several tasks have been explored by researchers in the past, including Information Extraction (Rahman et al., 2008; Uddin et al., 2019; Sharif et al., 2016), Machine Translation (Hasan et al., 2019; Anwar et al., 2009; Ismail et al., 2014), Named Entity Recognition (Banik and Rahman, 2018; Chaudhuri and Bhattacharya, 2008), Question Answering (Islam et al., 2016; Sarker et al., 2019; Kowsher et al., 2019), Sentiment Analysis (Uddin et al., 2019; Hassan et al., 2016; Alam et al., 2017), Spam and Fake Detection (Islam et al., 2019; Hussain et al., 2020) Annotator Bias Detection (Das et al., 2024), Hallucination Detection (Das et al., 2025) etc. Recently,

[1] Source: *List of languages by the number of native speakers*, Wikipedia, https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers (accessed on 23 May 2023)

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 91–106
December 23, 2025 ©2025 Association for Computational Linguistics

Kowsher et al. (2022) introduce Bangla-BERT, a monolingual BERT model for the Bangla language. Bhattacharjee et al. (2022) presented BanglaBERT – a BERT-based (Devlin et al., 2019) Bangla NLU model pre-trained on 27.5GB data. Transformer-based models have been frequently employed in Bangla NLP research, such as in abusive comment detection (Aurpa et al., 2022b), text classification (Alam et al., 2020), question answering (Aurpa et al., 2022a; Adnan and Anwar, 2022), machine translation (Dhar et al., 2022), image caption generation (Palash et al., 2022), etc. Researchers have also introduced different datasets such as BanglaNLG (Bhattacharjee et al., 2023), a comprehensive benchmark for evaluating natural language generation models in Bangla, a sentiment analysis dataset SentNoB (Islam et al., 2021), a dataset for Bangla fake news detection (Hossain et al., 2020).

Large Decoder (LLM) Based Generative Classification: Recent research has extensively studied the potential of large language models like ChatGPT, Flan, BLOOM, etc., for a wide range of applications. For example, researchers have shown the utility of ChatGPT in healthcare education (Sallam, 2023), programming bug solving (Surameery and Shakor, 2023), and machine translation (Jiao et al., 2023).

Classic Encoder Based Discriminative Classification: Various topic modeling-based approaches have been explored for zero-shot classification for the English language (Karmaker Santu et al., 2016). Similarly, Li et al. (2018); Zha and Li (2019) worked towards a data-less text classification. Veeranna et al. (2016), adopted pre-trained word embedding for measuring semantic similarity between a label and documents. Further endeavor has been spent on zero-shot learning using semantic embeddings by (Hascoet et al., 2019; Zhang et al., 2019; Xie and Virtanen, 2021).

Uniqueness of our Work: As mentioned above, the performances of recent LLMs on Bangla documents are mostly unknown. To address this research gap, we examined multiple state-of-the-art sentence encoders and large decoder-based LLMs for a well-defined NLP task, i.e., zero-shot multi-label classification, with an exclusive focus on Bangla documents. Our study lays a foundation for future research endeavors in this under-explored but important direction.

3 Problem Statement

We adopt the Definition-Wild 0SHOT-TC methodology proposed by Yin et al. (2019) and later explored by Sarkar et al. (2023a, 2022) for the English language, as defined below.

Definition 1. ***0SHOT-MLC:*** *Given a collection of documents denoted as $D = \{d_1, d_2, ..., d_n\}$, a user represented by U and a set of* ***user-defined*** *labels denoted as $L_U = \{l_1, l_2, ..., l_m\}$ provided in* ***real-time****, classify each document $d_i \in D$ with zero or more labels from L_U, without further fine-tuning.*

The 0SHOT-MLC employs embeddings of documents and labels, and the detailed methodology for embedding-based zero-shot-MLC is provided in Appendix A.1.

4 Experimental Setup

4.1 Dataset

We created a new benchmark corpus, BanglaNewsNet, by crawling a large collection of publicly available online news from a renowned Bangladeshi news website, https://www.prothomalo.com/. Each article here is already labeled with one or more labels by human annotators. For example, an article titled "মেসিকে রিয়াল বেতিসে চান তাঁর আর্জেন্টিনা দলের সতীর্থ" (Messi's Argentina team-mate wants him in Real Betis) is associated with labels "ফুটবল" (Football) and "আর্জেন্টিনা ফুটবল দল" (Argentina Football Team). We scraped the article titles, article texts and labels from the website and further cleaned the dataset by merging similar/repetitive labels into a single label, following approaches in (Sarkar and Karmaker, 2022). See Table 1 for an overview of the dataset.

Dataset Name	# of Articles	Avg. article length	Labels retained	Labels/ article
BanglaNewsNet	7245	≈2517 words	21	1.345

Table 1: An overview of the BanglaNewsNet dataset

4.2 Evaluation Metric

For evaluation, we report the Precision, Recall, and F_1 scores. Specifically, for each article, the model-inferred label(s) were compared against the list of "gold" label(s) to compute the true positive, false positive, and false negative statistics, which were then aggregated to compute the final Precision, Recall, and micro-averaged F_1 score.

4.3 Classic Encoders Vs. Large Decoders

For baselines, we implemented sentence similarity-based classification using LASER (Artetxe and Schwenk, 2019), LaBSE (Feng et al., 2020), and BanglaTransformer (Muhammad Rafsan Kabir et al., 2024). Additionally, we evaluated large decoder-based models from 13 LLM families, as listed in Table 2. For commercial LLMs (OpenAI, Google, Anthropic), we used their APIs, while open-source models were accessed via the Hugging Face Transformers library. Since APIs for BLOOM (Scao et al., 2022), FLAN-UL2 (Wei et al., 2021) and GPTNeoX (Black et al., 2022) were unavailable, we utilized their embeddings to enable direct comparison with traditional encoders. Beyond embedding-based approaches, we also evaluated LLMs on the Bangla Zero-Shot-MLC task using a prompt-based approach. Unlike embedding similarity methods, which require access to model representations, prompting enables direct multi-label classification using model-generated responses, making it a practical approach when working with black-box LLMs. The complete prompt used for Zero-Shot-MLC task is provided in Table 3.

LLM Family	Model
Google (Team et al., 2024)	gemini-1.5-pro (~200B) gemini-1.5-flash (8B) gemini-1.0-pro (~200B) gemma-2 (27B) gemma-2 (9B) gemma-2 (2B) flan-ul2 (20B)
OpenAI (Brown et al., 2020)	gpt-3.5-turbo-0613 (~20B) gpt-4o-mini-2024-07-18 (~20B) gpt-3 (175B)
EleutherAI (Black et al., 2022)	gpt-neox-20b (20B)
MetaAI Llama3 (Touvron et al., 2023)	Llama-3.1-8B (8B) Llama-3.2-3B (3B) meta-Llama-3.2-11B-Instruct (11B) Llama-3.3-70B-Instruct (70B)
BanglaLlama (Zehady et al., 2024)	BanglaLLama-3.1-8b (8B) BanglaLLama-3.2-3b (3B) BanglaLLama-3-11b (11B)
MistralAI (Jiang et al., 2024)	Mixtral-8x7B-Instruct-v0.1 (56B)
DeepSeek (DeepSeek-AI et al., 2025)	DeepSeek-R1-Distill-Llama-8B (8B) DeepSeek-R1-Distill-Qwen-14B (14B) DeepSeek-R1-Distill-Qwen-32B (32B)
AllenAI (OLMo et al., 2024)	OLMo-2-1124-7B-RM (7B)
Qwen (Yang et al., 2025)	Qwen1.5-72B (72B) Qwen2.5-7B-Instruct-1M (7B)
Gryphe (Gryphe)	MythoMax-L2-13b (13B)
UpStage (Kim et al., 2023)	SOLAR-10.7B-Instruct-v1.0 (10.7B)
Anthropic (Anthropic, 2024)	claude-3-haiku-20240307 (~20B)
BigScience (Scao et al., 2022)	bloom (176B)

Table 2: The list of Large Decoder families and their variants evaluated in Bangla Zero-Shot-MLC.

5 Results

This section presents the experimental results on the BanglaNewsNet dataset, evaluating models based on F_1 scores. Table 4 compares embedding-similarity methods using sentence encoders and three large decoder-based models. We utilized two label embedding approaches, such as *Label name + Keywords* and *Explicit-Mentions*, detailed in A.3. Based on the findings of Sarkar et al. (2023b), these embedding methods previously yielded the best results, leading us to focus exclusively on them in this study. Among encoders, LaBSE performed best, achieving an F_1 score of around 40%, while BanglaTransformer performed slightly lower. For large decoder-based models, Flan-UL2, BLOOM, and GPT-NeoX failed to achieve strong results in 0SHOT-MLC on Bangla articles. Although some models exhibited high recall, their low precision made the results less meaningful. Notably, no LLM surpassed LaBSE in embedding-based 0SHOT-MLC, highlighting the limitations of current LLMs in handling their linguistic complexities.

5.1 LLM Performance Across Model Sizes

Table 5 presents the performance of LLMs categorized by parameter size. Below is a summary of key observations for each model category:

Small Models (<8B). OLMo-7B-Instruct (Groeneveld et al., 2024) and Qwen-2.5-7B-Instruct (Team, 2024b) outperform other models in this category, yet their F_1 scores (0.381 and 0.351, respectively) remain low. Surprisingly, none of the LLMs in this group exceeds the classic encoder LaBSE (F_1: 0.400), highlighting their limitations in the context of Bangla Zero-Shot-MLC.

Mid-Sized Models (8B - 10B). Gemini-1.5-Flash (8B) (Team et al., 2024) (F_1: 0.571) outperforms larger models, surpassing Gemma-2-9B (Team, 2024a) (F_1: 0.544) and DeepSeek-R1-Distill-8B (Guo et al., 2025) (F_1: 0.470).

Intermediate Models (10B - 15B). MythoMax-L2 (13B) (Gryphe) underperforms significantly (F_1: 0.187). Llama 3.2 (11B) (Touvron et al., 2023) and BanglaLlama 3.2 (11B) (Zehady et al., 2024) achieve close scores (0.513 and 0.481), showing minimal benefits from language-specific tuning. MythoMax-L2 (13B) (Gryphe) underperforms significantly (F_1: 0.187). Llama 3.2 (11B) (Touvron et al., 2023) and BanglaLlama 3.2 (11B) (Zehady et al., 2024) achieve close

Prompt Design	
System setup	
The AI assistant has been designed to understand and categorize user input by the given labels. When processing user input, the assistant must predict the labels from one of the following pre-defined options: 'চাকরিবাকরি' (Job market), 'করোনাভাইরাস' (Coronavirus), 'চলচ্চিত্র ও তারকা' (Movies and celebrities), 'স্বাস্থ্য' (Health), 'ব্যাংক' (Banking), 'অর্থনীতি' (Economy), 'শিক্ষা' (Education), 'প্রাকৃতিক দুর্যোগ' (Natural disasters), 'আইন ও আদালত' (Law and justice), 'কূটনীতি' (Politics), 'শিল্প ও বাণিজ্য' (Industry and commerce), 'ভ্রমণ' (Travel), 'নকশা' (Design), 'ফুটবল' (Football), 'খাবারদাবার' (Food and dining), 'দেশ ও রাজনীতি' (Country and politics), 'আন্তর্জাতিক' (International), 'দেশের খবর' (Country news), 'রাশিয়া ইউক্রেন সংঘাত' (Russia-Ukraine conflict), 'ক্রিকেট' (Cricket), 'নারী' (Women). It is essential to note that an article may have multiple labels. If the user input is not relevant with any labels, the assistant should print nothing, indicating that the input does not align with the available categories. The agent MUST response with the following json format: {**“Labels”: [“List of labels”**]}	
User	**Taking into account the given Bangla article** {ব্যাংক এশিয়া ২০১৪ সালে ব্যাংকিং সেবার বাইরে থাকা বিপুল জনগোষ্ঠীকে ব্যাংকিং সেবায় আনতে এজেন্ট ব্যাংকিং সেবা চালু করে। বর্তমানে রাষ্ট্রমালিকানাধীন ও বেসরকারি মিলিয়ে ৩১টি ব্যাংক এ সেবা দিচ্ছে। বর্তমানে এজেন্ট ব্যাংকিং সেবা গ্রহণকারীর সংখ্যা দেড় কোটির বেশি। এর মধ্যে ব্যাংক এশিয়ার গ্রাহক ৫৫ লাখের বেশি। এসব গ্রাহকের ৯২ শতাংশই গ্রামীণ জনগোষ্ঠী। আবার ৬২ শতাংশ গ্রাহকই নারী। সারা দেশে ব্যাংক এশিয়ার ৫ হাজার ৪০০-এর বেশি এজেন্ট আউটলেট রয়েছে, যাদের মধ্যে নারী এজেন্ট ৫৪০ জন। ..... (In 2014, Bank Asia introduced agent banking services to bring banking services to a large population that was outside the reach of traditional banking. At present, a total of 31 banks, including both public and private, are providing these services. The number of users availing agent banking services has surpassed 140 million. Among them, Bank Asia has more than 5.5 million customers, of which 92% are from rural areas. Furthermore, 62% of the customers are women. Throughout the country, there are more than 5,400 agent outlets of Bank Asia, including 540 outlets managed by female agents.....)}, predict the category or labels of this article from the list of mentioned labels.
Assistant	{“**Labels**”: “ব্যাংক” (Banking), “শিল্প ও বাণিজ্য” (Industry and commerce)}
Directive: Taking into account the given Bangla article {article text}, predict the category or labels of this article from the list of mentioned {labels}. Please remember to only respond in the predefined JSON format without any additional information.	

Table 3: Prompt design details for the Zero-Shot-MLC task on BanglaNewsNet.

scores (0.513 and 0.481), showing minimal benefits from language-specific tuning. Although BanglaLlama is fine-tuned on Bangla corpora, it fails to outperform the base Llama model. A plausible explanation is that because BanglaLlama was trained on specific data sources such as WikiBangla rather than on task-oriented datasets. As a result, its adaptation improves linguistic alignment but not reasoning ability.

Larger Models (15B - 50B). GPT-4o Mini (20B) (Brown et al., 2020) and Claude-3 Haiku (20B) (Anthropic, 2024) perform well (F_1: 0.588, 0.540). Gemma-2 (27B) (Team, 2024a) leads this category (F_1: 0.593). DeepSeek R1 Distill (32B) (Guo et al., 2025) also performs well (F_1: 0.550), reinforcing the effectiveness of model distillation.

Largest Models (>50B). Gemini-1.5-Pro (Team et al., 2024) and Gemini-1.0-Pro (Team et al., 2023) achieve the highest F_1 scores (0.616, 0.589). Despite their size, Mixtral-56B-Instruct (F_1: 0.305) (Jiang et al., 2024), Qwen-1.5 (Yang et al., 2025) (F_1: 0.429), and GPT-3.5 (Brown et al., 2020) (F_1: 0.417) significantly underperformed.

Overall, the Gemini models demonstrate strong performance on this task. Gemini-1.5-Pro achieves the highest F_1 score (0.616), driven largely by its high recall (0.918). Although this indicates that the model successfully identifies most relevant instances, such unusually high recall—paired with comparatively low precision (0.463)—suggests that the model may be over-generating labels, capturing correct cases, but also introducing many false positives. In contrast, Gemini-1.0-Pro achieves the second-highest F_1 score (0.589) with notably strong precision (0.796), reflecting more conservative and accurate predictions with fewer false alarms. A full overview

Classical Encoder Models						
	LASER		**LaBSE**		**BanglaTransformer**	
Label Embedding ->	**Label+KW**	**Expl.-Ment.**	**Label+KW**	**Expl.-Ment.**	**Label+KW**	**Expl.-Ment.**
F1 Score	0.267	0.305	0.354	0.404	0.334	0.384
Large Decoder Models						
	Flan-UL2		**Bloom**		**GPTNeoX**	
Label Embedding ->	**Label+KW**	**Expl.-Ment.**	**Label+KW**	**Expl.-Ment.**	**Label+KW**	**Expl.-Ment.**
F1 Score	0.234	0.241	0.329	0.341	0.345	0.357

Table 4: F1 score comparison of embedding based Zero-Shot-MLC across classical encoders and large decoders.

<8B Parameters		**8B - 10B Parameters**		**10B - 15B Parameters**		**15B - 50B Parameters**		**>50B Parameters**	
Model	F_1	**Model**	F_1	**Model**	F_1	**Model**	F_1	**Model**	F_1
Llama 3.2 (3B)	**0.320**	Llama 3.1 (8B)	**0.476**	Llama 3.2 (11B)	**0.513**	GPT 3.5 Turbo (~20B)	**0.470**	Llama 3.3 (70B)	**0.558**
BanglaLlama 3.2 (3B)	**0.323**	BanglaLlama 3.1 (8B)	**0.424**	BanglaLlama 3.2 (11B)	**0.481**	GPT 4o Mini (~20B)	**0.588**	Gemini 1.0 Pro (~200B)	**0.589**
Gemma 2 (2B)	**0.280**	Gemini 1.5 Flash (8B)	**0.571**	MythoMax L2 (13B)	**0.187**	Gemma 2 (27B)	**0.593**	Gemini 1.5 Pro (~200B)	**0.616**
OLMo 7B Instruct (7B)	**0.380**	Gemma 2 (9B)	**0.544**	SOLAR 10.7B Instruct (10.7B)	**0.452**	Claude 3 Haiku (~20B)	**0.540**	Mixtral 56B Instruct (56B)	**0.305**
Qwen 2.5 7B Instruct (7B)	**0.351**	DeepSeek R1 Distill (8B)	**0.470**	DeepSeek R1 Distill (14B)	**0.495**	DeepSeek R1 Distill (32B)	**0.550**	Qwen 1.5 (72B)	**0.429**
								GPT 3.5 (175B)	**0.537**

Table 5: F1 score comparison of prompting-based approaches across models with varying parameter sizes.

of the precision-recall trade-off across all evaluated models is provided in Table 8 in the Appendix.

Interesting Findings.

- **Top-performing models.** Gemini-1.5-Pro (200B, F_1: 0.616) and Gemma-2 (27B, F_1: 0.593) achieve the highest scores.
- **Bigger isn't always better.** Gemini-1.5-Flash (8B) outperforms larger models like DeepSeek R1 Distill (32B), Mixtral (56B), and GPT-3.5 (175B).
- **Encoders still hold value.** The classic encoder LaBSE (F_1: .404) surprisingly outperforms all 8B-based models, reinforcing its effectiveness in zero-shot-MLC.
- **LLMs struggle with low-resource languages.** Despite strong performance in English, most models fail to generalize well in Bangla, exposing a multilingual gap.

6 Discussion and Conclusion

Our study takes an important step toward evaluating state-of-the-art models for low-resource languages, with a focus on Bangla. We benchmark classic encoders and large decoder-based LLMs for Zero-Shot Multi-Label Classification (Zero-Shot-MLC), uncovering key limitations in both approaches. While large decoders perform well in high-resource languages, their effectiveness in Bangla remains inconsistent, with most failing to achieve reliable classification despite their scale. Classical encoders, though more stable, did not show significant improvement, indicating that neither approach is fully optimized for Bangla and exposing critical gaps in multilingual adaptation.

This work addresses a crucial research gap and also reinforces the need for tailored approaches to handle the complexity of morphologically rich, low-resource languages like Bangla. By evaluating both encoders and decoder-based models, we contribute to improve multilingual NLP for regional languages.

Limitations

As a limitation, it is worth mentioning that we had restricted access to several large language models (LLMs) such as LaMDA (137B), Jurassic-1 (178B), ERNIE 3.0 Titan (260B), Gopher (280B), GPT-4, Megatron Turing NLG (530B) and DeepSeek-R1 (671B), which limited their inclusion in our experiments. Additionally, while models like ChatGPT were available through an API, their usage was constrained by cost, which scales with model size, making large-scale evaluations financially challenging. Our computational resources were also a limiting factor, as we could only run open-source models up to 70 billion parameters, restricting our ability to test larger models natively. As a result, for models exceeding this threshold, we had to rely solely on API-based access, further increasing cost constraints. Furthermore, hardware limitations affected our ability to fine-tune models efficiently, limiting our exploration of task-specific optimizations. Another constraint was dataset diversity. Our experiments were conducted on a single dataset, and broader evaluations are necessary to determine whether our findings generalize across different domains.

7 Ethical Considerations

In this study we used publicly available news articles to prepare our dataset. We didn't include any personal or sensitive information in our dataset. All the articles we collected are already published online for public reading, and we used them only for research and analysis. We evaluated different language models which may carry hidden biases. We did not fine-tune any commercial model with private data, and all API-based evaluations followed the provider's usage policies. As our experiment focused on Bangla, a low-resource language, models may show uneven performance across different topics and lead to misclassifications. We encouraged responsible use of our findings and advise that any downstream use should consider these limitations.

References

Md Abdul Mazid Adnan and Md Musfique Anwar. 2022. Progressive guidance categorization using transformer-based deep neural network architecture. In *Hybrid Intelligent Systems: 21st International Conference on Hybrid Intelligent Systems (HIS 2021), December 14–16, 2021*, volume 420, page 344. Springer Nature.

Md. Habibul Alam, Md-Mizanur Rahoman, and Md. Abul Kalam Azad. 2017. Sentiment analysis for bangla sentences using convolutional neural network. In *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–6.

Tanvirul Alam, Akib Khan, and Firoj Alam. 2020. Bangla text classification using transformers. *arXiv preprint arXiv:2011.04446*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. Accessed: 2025-02-14.

Md Musfique Anwar, Mohammad Zabed Anwar, and M Al-Amin Bhuiyan. 2009. Syntax analysis and machine translation of bangla sentences. *International Journal of Computer Science and Network Security*, 9(8):317–326.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Trans. Assoc. Comput. Linguistics*, 7:597–610.

Tanjim Taharat Aurpa, Richita Khandakar Rifat, Md Shoaib Ahmed, Md Musfique Anwar, and ABM Shawkat Ali. 2022a. Reading comprehension based question answering system in bangla language with transformer-based learning. *Heliyon*, 8(10):e11052.

Tanjim Taharat Aurpa, Rifat Sadik, and Md Shoaib Ahmed. 2022b. Abusive bangla comments detection on facebook using transformer-based deep learning models. *Social Network Analysis and Mining*, 12(1):24.

Nayan Banik and Md. Hasan Hafizur Rahman. 2018. Gru based named entity recognition system for bangla online newspapers. In *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, pages 1–6.

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.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bidyut Baran Chaudhuri and Suvankar Bhattacharya. 2008. An experiment on automatic detection of named entities in Bangla. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*.

Amit Das, Md Najib Hasan, Souvika Sarkar, Zheng Zhang, Fatemeh Jamshidi, Tathagata Bhattacharya, Nilanjana Raychawdhury, Dongji Feng, Vinija Jain, and Aman Chadha. 2025. Investigating hallucination in conversations for low resource languages. *arXiv preprint arXiv:2507.22720*.

Amit Das, Zheng Zhang, Najib Hasan, Souvika Sarkar, Fatemeh Jamshidi, Tathagata Bhattacharya, Mostafa Rahgouy, Nilanjana Raychawdhary, Dongji Feng, Vinija Jain, et al. 2024. Investigating annotator bias in large language models for hate speech detection. *arXiv preprint arXiv:2406.11109*.

Daya Guo DeepSeek-AI, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Argha Chandra Dhar, Arna Roy, Md Ahsan Habib, MAH Akhand, and N Siddique. 2022. Transformer deep learning model for bangla–english machine translation. In *Proceedings of 2nd International Conference on Artificial Intelligence: Advances and Applications: ICAIAA 2021*, pages 255–265. Springer.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Gryphe. Mythomax-l2-13b-turbo. `https://deepinfra.com/Gryphe/MythoMax-L2-13b-turbo/api`. Accessed: 2025-02-14.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Md. Arid Hasan, Firoj Alam, Shammur Absar Chowdhury, and Naira Khan. 2019. Neural machine translation for the bangla-english language pair. In *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, pages 1–6.

Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. 2019. Semantic embeddings of generic objects for zero-shot learning.

EURASIP Journal on Image and Video Processing, 2019(1):1–14.

Asif Hassan, Mohammad Rashedul Amin, Abul Kalam Al Azad, and Nabeel Mohammed. 2016. Sentiment analysis on bangla and romanized bangla text using deep recurrent models. In *2016 International Workshop on Computational Intelligence (IWCI)*, pages 51–56.

Md Zobaer Hossain, Md Ashraful Rahman, Md Saiful Islam, and Sudipta Kar. 2020. BanFakeNews: A dataset for detecting fake news in Bangla. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2862–2871, Marseille, France. European Language Resources Association.

Md Gulzar Hussain, Md Rashidul Hasan, Mahmuda Rahman, Joy Protim, and Sakib Al Hasan. 2020. Detection of bangla fake news using mnb and svm classifier. In *2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pages 81–85.

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.

Md. Aminul Islam, Md. Fasihul Kabir, Khandaker Abdullah-Al-Mamun, and Mohammad Nurul Huda. 2016. Word/phrase based answer type classification for bengali question answering system. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 445–448.

Tanvirul Islam, Subhenur Latif, and Nadim Ahmed. 2019. Using social networks to detect malicious bangla text content. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pages 1–4.

Sabir Ismail, M. Shahidur Rahman, and Md Abdullah Al Mumin. 2014. Developing an automated bangla parts of speech tagged dictionary. *16th Int'l Conf. Computer and Information Technology*, pages 355–359.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? a preliminary study. *arXiv preprint arXiv:2301.08745*.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2016. Generative feature language models for mining implicit features from customer reviews. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 929–938.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. 2023. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*.

M. Kowsher, Abdullah As Sami, Nusrat Jahan Prottasha, Mohammad Shamsul Arefin, Pranab Kumar Dhar, and Takeshi Koshiba. 2022. Bangla-bert: Transformer-based efficient model for transfer learning and language understanding. *IEEE Access*, 10:91855–91870.

Md. Kowsher, M M Mahabubur Rahman, Sk Shohorab Ahmed, and Nusrat Jahan Prottasha. 2019. Bangla intelligence question answering system based on mathematics and statistics. In *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, pages 1–6.

Ximing Li, Changchun Li, Jinjin Chi, Jihong Ouyang, and Chenliang Li. 2018. Dataless text classification: A topic modeling approach with document manifold. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 973–982, New York, NY, USA. Association for Computing Machinery.

Muhammad Rafsan Kabir, Md. Mohibur Rahman Nabil, and Mohammad Ashrafuzzaman Khan. 2024. Banglaembed: Efficient sentence embedding models for a low□resource language using cross□lingual distillation techniques. *arXiv preprint arXiv:2411.15270*. Preprint.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2024. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*.

Md Aminul Haque Palash, Md Abdullah Al Nasim, Sourav Saha, Faria Afrin, Raisa Mallik, and Sathishkumar Samiappan. 2022. Bangla image caption generation through cnn-transformer based encoder-decoder network. In *Proceedings of International Conference on Fourth Industrial Revolution and Beyond 2021*, pages 631–644. Springer.

Mohammad Osiur Rahman, Fouzia Asharf Mousumi, Edgar Scavino, Aini Hussain, and Hassan Basri. 2008. Real time road sign recognition system using artificial neural networks for bengali textual information box. In *2008 International Symposium on Information Technology*, volume 2, pages 1–8.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Malik Sallam. 2023. Chatgpt utility in healthcare education, research, and practice: systematic review on the promising perspectives and valid concerns. In *Healthcare*, volume 11, page 887. MDPI.

Souvika Sarkar, Dongji Feng, and Shubhra Kanti Karmaker Santu. 2022. Exploring universal sentence encoders for zero-shot text classification. 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 135–147, Online only. Association for Computational Linguistics.

Souvika Sarkar, Dongji Feng, and Shubhra Kanti Karmaker Santu. 2023a. Zero-shot multi-label topic inference with sentence encoders and LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16218–16233, Singapore. Association for Computational Linguistics.

Souvika Sarkar, Dongji Feng, and Shubhra Kanti Karmaker_Santu. 2023b. Zero-shot multi-label topic inference with sentence encoders and llms. Association for Computational Linguistics.

Souvika Sarkar and Shubhra Kanti Karmaker. 2022. Concept annotation from users perspective: A new challenge. In *Companion proceedings of the web conference 2022*, pages 1180–1188.

Sourav Sarker, Syeda Tamanna Alam Monisha, and Md Mahadi Hasan Nahid. 2019. Bengali question answering system for factoid questions: A statistical approach. In *2019 International Conference on Bangla*

Speech and Language Processing (ICBSLP), pages 1–5.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

S M A Sharif, Nabeel Mohammed, Nafees Mansoor, and Sifat Momen. 2016. A hybrid deep model with hog features for bangla handwritten numeral classification. In *2016 9th International Conference on Electrical and Computer Engineering (ICECE)*, pages 463–466.

Nigar M Shafiq Surameery and Mohammed Y Shakor. 2023. Use chat gpt to solve programming bugs. *International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290*, 3(01):17–22.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Gemma Team. 2024a. Gemma.

Qwen Team. 2024b. Qwen2. 5: A party of foundation models. *Qwen (Sept. 2024). url: https://qwenlm. github. io/blog/qwen2*, 5.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Abdul Hasib Uddin, Sumit Kumar Dam, and Abu Shamim Mohammad Arif. 2019. Extracting severe negative sentence pattern from bangla data via long short-term memory neural network. In *2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, pages 1–6.

Sappadla Prateek Veeranna, Jinseok Nam, EL Mencía, and J Furnkranz. 2016. Using semantic similarity for multi-label zero-shot classification of text documents. In *Proceeding of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges, Belgium: Elsevier*, pages 423–428.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Huang Xie and Tuomas Virtanen. 2021. Zero-shot audio classification via semantic embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1233–1242.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, et al. 2025. Qwen2. 5-1m technical report. *arXiv preprint arXiv:2501.15383*.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3912–3921. Association for Computational Linguistics.

Abdullah Khan Zehady, Safi Al Mamun, Naymul Islam, and Santu Karmaker. 2024. Bongllama: Llama for bangla language. *arXiv preprint arXiv:2410.21200*.

Daochen Zha and Chenliang Li. 2019. Multi-label dataless text classification with topic modeling. *Knowledge and Information Systems*, 61(1):137–160.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1031–1040. Association for Computational Linguistics.

A Appendix

A.1 *Encoder & LLM* Based 0-shotMLC

Here, we describe the steps used in our 0-shot-MLC approach.

1. Input Document: The end user provides the article text, custom-defined labels, and a set of keywords (optional).
2. Embedding Generation: The article text, labels, and keywords are transformed into rich embedding by leveraging the power of language models and encoding methods, capturing the essence of their content.
   - **Article Embedding**: We embed the entire article with sentence encoders and LLMs in a single shot.
   - **Label Embedding**: We adopted two different ways for target label embedding: 1) *Label + Keyword*- Label embedding using label name and keywords, 2) *Explicit-Mentions*- Label embedding using article-text which contains explicit mentions of label names.

   The details of these embeddings have been discussed in the Appendix A.2, A.3. While there are certainly other embedding methods possible, based on the findings of the paper (Sarkar et al., 2023b), these embedding combinations worked best previously, and hence, we focused only on the above embedding type.
3. Threshold-based Label Assignment: Next, we quantify the cosine similarity between the article embedding and the label embeddings. Labels are assigned to the article based on a specified threshold, indicating the presence of a strong association. By experimenting with different threshold values (ranging from 0-1), a comprehensive analysis is conducted.
4. Zero-Shot multi-label classification: The outcome of this classifier is the prediction of relevant label(s) for the given article.

A.2 "Entire Article" based embedding

Encode the entire article using sentence encoders or LLMs in a single shot, including articles that are long paragraphs and consist of more than one sentence.

A.3 Label Embedding Approaches

We have used 2 different approaches for computing label embedding. The consecutive sections discuss about different procedures for generating label embedding.

A.3.1 "Label name + Keywords" based embedding

Encode both label name and keywords, then average all embeddings to generate the final label embedding.

A.3.2 "Explicit-Mentions" based embedding

First, we extract all the articles explicitly mentioning the label/phrase using algorithm 1 for all labels. Then, for each label, we generate embeddings of all articles that are explicitly annotated/classified with that label, then average them to obtain the ultimate label embedding.

A.4 Baseline Sentence Encoders

This section presents a bird's-eye view of the sentence encoders and large language models we have used for our experiments.

- **Language-Agnostic SEntence Representations (LASER)**: LASER is

Algorithm 1 Article Annotation using Explicit Mention

```
Input: Article text, Label names, and Keywords
Output: Articles annotated with explicit Label
for each article text do
    check whether the label name or set (at least 3) of the informative keywords are present or not in the corresponding article text
    if present then annotate the article with the explicit label
    end if
end for
```

a sentence encoding model that generates language-agnostic representations (Artetxe and Schwenk, 2019). It is capable of encoding sentences from multiple languages into fixed-length vectors, enabling cross-lingual tasks and multilingual applications.

- **Language-agnostic BERT Sentence Embedding (LaBSE)**: LaBSE (Feng et al., 2020) is a language-agnostic model based on the BERT architecture. It provides sentence embeddings that capture the semantic meaning of sentences across different languages. LaBSE allows for cross-lingual understanding and transfer-learning tasks.
- **Bangla sentence embedding transformer**: This Bangla sentence transformer (Muhammad Rafsan Kabir et al., 2024) is specifically designed for the Bangla language. It utilizes a transformer-based architecture to encode Bangla sentences into meaningful representations, enabling various NLP tasks in Bangla text analysis. It was trained on 2,50,000 Bangla sentences(wiki) by Sentence-Transformer. This work is inspired by Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (Reimers and Gurevych, 2019) technique.

A.5 Large Language Models

- **BLOOM**: Scao et al. (2022) introduce BLOOM, a massive language model with 176 billion parameters. BLOOM is trained on 46 natural languages and 13 programming languages and is the result of a collaborative effort involving hundreds of researchers. BLOOM is a causal language model trained to predict the next token in a sentence. This approach has been found effective in capturing reasoning abilities in large language models. BLOOM uses a Transformer architecture composed of an input embedding layer, 70 Transformer blocks, and an output language-modeling layer. The sequential operation of predicting the next token involves passing the input tokens through each of the 70 BLOOM blocks. To prevent memory overflow, only one block is loaded into RAM at a time. The word embeddings and output language-modeling layer can be loaded on demand from disk.
- **GPT-NeoX**: The GPT-NeoX-20B paper, authored by the Black et al. (2022), introduces an architecture similar to GPT-3 but with notable differences. They utilize rotary positional embeddings for token position encoding instead of learned embeddings and parallelize the attention and feed-forward layers, resulting in a 15% increase in throughput. Unlike GPT-3, GPT-NeoX-20B exclusively employs dense layers. The authors trained GPT-NeoX-20B using EleutherAI's custom codebase (GPT-NeoX) based on Megatron and DeepSpeed, implemented in PyTorch. To address computational limitations, the authors reused the hyperparameters from the GPT-3 paper. In their evaluation, the researchers compared GPT-NeoX-20B's performance to their previous model, GPT-J-6B, as well as Meta's FairSeq 13B and different sizes of GPT-3 on various NLP benchmarks, including LAMBADA, WinoGrande, HendrycksTest, and the MATH dataset. While improvements were desired for NLP tasks, GPT-NeoX-20B exhibited exceptional performance in science and math tasks.
- **Google (Gemini & Flan)**: (Team et al., 2024) introduced Gemini-1.5 Pro and 1.0 Pro models containing powerful multimodal capabilities, allowing them to process text, images, audio, and video at the

same time. This makes them highly effective for real-time AI tasks like reasoning and code generation. With an optimized transformer-decoder and sparse mixture-of-experts techniques, they enhance efficiency while handling complex inputs. Their long-context attention feature also helps them retain and process longer pieces of information more effectively. Meanwhile, the Gemma family (2B, 9B, and 27B) is built for flexible deployment, whether on-device or in the cloud, making it ideal for fast, low-latency applications. Flan-UL2, an improved version of Flan-T5, uses Mixture-of-Denoisers (MoD) pre-training to push the boundaries of NLP tasks like classification, reasoning, and question answering. With 20 billion parameters, it outperforms models like T5 and GPT-3, excelling in zero-shot learning and chain-of-thought reasoning while achieving top results on major NLP benchmarks.

- **OpenAI (GPT Series)**: According to (Brown et al., 2020) GPT-4o, GPT-3.5 Turbo, and GPT-3 represent major leaps in AI language models. GPT-4o is the most advanced, handling text, images, and audio at the same time with better speed and accuracy, making it great for real-time AI applications like chat-bots and coding assistants. GPT-3.5 Turbo is designed for efficiency, balancing cost and performance, which makes it popular for business AI tools and content generation. GPT-3, with its 175 billion parameters, was a game-changer in AI, setting the stage for today's models with its strong language understanding and reinforcement learning for better alignment with human values.
- **MetaAI (Llama3)**: (Touvron et al., 2023) mentioned Meta's Llama3 models Llama-3.1-8B, Llama-3.2-3B, and Llama-3.3-70B are designed for efficient, cost-effective AI deployment. They use adaptive tokenization and transformer pruning to reduce computational demands while maintaining strong performance. The Llama-3.2-11B Vision-Instruct model expands Meta's work in multi-modal AI, integrating visual and language reasoning for applications like computer vision, medical imaging, and smart assistants. By open-sourcing its models, Meta promotes collaborative AI development and innovation in decentralized AI systems. With real-time processing capabilities, Llama3 models stand out for their speed and energy efficiency, making them ideal for on-premise AI, embedded systems, and low-latency conversational tools.
- **BanglaLlama**: (Zehady et al., 2024) introduced BanglaLlama which is a groundbreaking initiative aimed at improving NLP for the Bangla language, addressing the lack of high-quality AI models for low-resource languages. With models ranging from 3B to 11B parameters, it is trained specifically on Bangla text to better capture linguistic nuances, cultural context, and accuracy. By using advanced tokenization and dataset augmentation, BanglaLlama excels in translation, content creation, and conversational AI. Its fine-tuned instruction-following capabilities help make AI more inclusive, ensuring non-English languages are better represented in global AI advancements. Beyond language processing, BanglaLlama also plays a crucial role in reducing biases, adapting to different Bangla dialects, and preserving indigenous languages in AI systems.
- **MistralAI**: (Jiang et al., 2024) represents Mixtral-8x7B as a groundbreaking Mixture of Experts (MoE) model, significantly reducing computational complexity while enhancing inference efficiency. Unlike conventional transformer architectures, Mixtral activates only a subset of its parameters per forward pass, reducing memory footprint and improving scalability. This innovative architecture allows highly efficient parallelized computation, making it a top contender for large-scale enterprise AI applications, real-time NLP solutions, and multilingual text generation. Mis-

tralAI's MoE-based LLMs are widely recognized for their superior speed-to-accuracy trade-offs, positioning them as one of the most energy-efficient large-scale models in the industry.

- **DeepSeek**: (DeepSeek-AI et al., 2025) mentioned DeepSeek-R1 series, is designed for maximum efficiency, using advanced knowledge distillation to maintain strong performance with fewer parameters. With techniques like quantization and structured pruning, these models deliver fast inference speeds and low-latency processing, making them perfect for AI applications with limited resources. Ideal for real-time NLP, enterprise automation, and AI assistants, DeepSeek-R1 ensures quick responses while keeping computational demands low, making it a great choice for businesses and developers focused on efficiency.
- **AllenAI**: (OLMo et al., 2024) designed OLMo-2-7B with a strong focus on explainable AI (XAI) and interpretability. Unlike traditional black-box AI models, it incorporates features like attention transparency and explainability layers, ensuring clearer insights into how it processes information. Optimized for research in linguistics, AI ethics, and decision-making, OLMo is a top choice for academics, policymakers, and those building transparent AI systems that prioritize human understanding and trust.
- **Alibaba**: According to (Yang et al., 2025) Alibaba's Qwen series, including Qwen1.5-72B and Qwen2.5-7B, is built for enterprise applications, specializing in complex reasoning, structured NLP, and industry-specific adaptability. These models are fine-tuned for tasks like financial analysis, healthcare AI, and multilingual document processing, making them highly versatile for business use. With strong instruction tuning, the Qwen models excel at knowledge-intensive tasks, delivering high factual accuracy and outperforming many mainstream models in industry automation and specialized AI applications.
- **Gryphe**: (Gryphe) designed Gryphe's MythoMax-L2-13B for creative content generation, interactive storytelling, and narrative coherence. This model incorporates fine-tuned stylistic awareness and logical consistency, making it a preferred choice for conversational AI, virtual assistants, and AI-driven journalism.
- **UpStage**: (Kim et al., 2023) designed SOLAR-10.7B for flexibility, excelling in instruction tuning, few-shot learning, and adapting to various AI tasks. With a structured fine-tuning approach, it performs exceptionally well in areas like legal text interpretation, scientific research support, and summarizing complex documents with context and accuracy.
- **Anthropic**: (Anthropic, 2024) built Claude-3 Haiku with a strong focus on ethical AI, safety, and human-guided learning (RLHF). It excels at maintaining context, ensuring fairness, and delivering reliable real-time conversations, making it a great fit for critical applications in healthcare, governance, and compliance-focused AI.

A.6 Precision-Recall Trade-Off in LLMs for Bangla

The performance analysis of large language models (LLMs) for zero-shot multi-label classification (MLC) in Bangla highlights several important aspects, particularly the trade-offs between precision and recall, which we have explored in the appendix due to space constraints.

A.6.1 Sentence-Encoder Models

Looking at the baseline sentence-encoder-based approaches (Table 6), we see that BanglaTransformer achieves the highest $\boldsymbol{F_1}$ score (0.334), but with an evident imbalance between precision and recall. This pattern is also observed in LASER and LABSE, where recall is consistently higher than precision. While higher recall means the model retrieves more relevant labels, it also increases the number of false positives, which is a common issue in low-resource languages where high-quality training data is limited.

Topic+Keywords Based Label Embedding								
LASER			LaBSE			BanglaTransformer		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.162	0.750	0.267	0.282	0.477	**0.354**	0.224	0.648	0.334
Explicit-Mention Based Label Embedding								
LASER			LaBSE			BanglaTransformer		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.193	0.724	0.305	0.300	0.617	**0.404**	0.276	0.635	0.384

Table 6: Performance comparison of baseline sentence encoder-based approaches.

Topic+Keywords Based Label Embedding								
FLAN-UL2			BLOOM			GPT-NeoX		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.135	0.890	0.234	0.231	0.574	0.329	0.235	0.634	0.345
Explicit-Mention Based Label Embedding								
FLAN-UL2			BLOOM			GPT-NeoX		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.144	0.742	0.241	0.232	0.642	0.341	0.241	0.675	0.357

Table 7: Performance comparison of different large language models.

Performance Comparison of LLMs with Varying Parameter Sizes				
Parameter Size	**Model**	**Precision**	**Recall**	F_1
<8B	Llama 3.2 (3B)	0.203	0.710	0.320
	BanglaLlama 3.2 (3B)	0.219	0.595	0.323
	Gemma 2 (2B)	0.164	0.942	0.280
	OLMo 7B Instruct (7B)	0.365	0.410	0.380
	Qwens 2.5 7B Instruct (7B)	0.218	0.906	0.351
8B to 10B	Llama 3.1 (8B)	0.325	0.886	0.476
	BanglaLlama 3.1 (8B)	0.289	0.790	0.424
	Gemini 1.5 Flash (8B)	0.416	0.905	0.571
	Gemma 2 (9B)	0.388	0.913	0.544
	DeepSeek R1 Distill (8B)	0.343	0.782	0.470
10B to 15B	Llama 3.2 (11B)	0.359	0.895	0.513
	BanglaLlama 3.2 (11B)	0.336	0.849	0.481
	MythoMax L2 (13B)	0.128	0.347	0.187
	SOLAR 10.7B Instruct (10.7B)	0.342	0.650	0.452
	DeepSeek R1 Distill (14B)	0.350	0.850	0.495
15B to 50B	GPT 3.5 Turbo (~20)	0.350	0.741	0.470
	GPT 4o Mini (~20B)	0.439	0.889	0.588
	Gemma 2 (27B)	0.450	0.833	0.593
	Claude 3 Haiku (~20B)	0.402	0.810	0.540
	DeepSeek R1 Distill (32B)	0.418	0.804	0.550
50B>	Llama 3.3 (70B)	0.401	0.921	0.558
	Gemini 1.0 Pro (~200B)	0.796	0.468	0.589
	Gemini 1.5 Pro (~200B)	0.463	0.918	0.616
	Mixtral 56B Instruct (56B)	0.194	0.630	0.305
	Qwens 1.5 (72B)	0.353	0.540	0.429
	GPT 3.5 (175B)	0.515	0.573	0.537

Table 8: Performance comparison of prompting-based approaches across models with varying parameter sizes.

A.6.2 Large-Scale Generative Models

Shifting to large-scale generative models (Table 7), we observe a noticeable improvement in $\boldsymbol{F_1}$ scores, with GPT-NeoX (Black et al., 2022) ($\boldsymbol{F_1}$: 0.357) outperforming BLOOM and FLAN-UL2 (Wei et al., 2021). However, GPT-NeoX's (Black et al., 2022) precision (0.241) remains significantly lower than its recall (0.675), reinforcing a major trend seen in generative models: they tend to be highly recall-biased, favoring coverage over accuracy. This can be attributed to their training approach, which optimizes for broad knowledge retrieval rather than precise classification. While this makes them effective for open-ended generation tasks, it poses a challenge for multi-label classification, where specificity is crucial. A recall-heavy approach may work well in some cases, such as medical document classification, where missing a critical label could be costly, but for general-purpose classification tasks, such a model could introduce significant noise. This reinforces the need for additional fine-tuning or hybrid methods to improve precision without sacrificing recall. Examining LLMs across different parameter sizes (Table 8), we see a clear scaling trend, where larger models generally achieve higher $\boldsymbol{F_1}$ scores but with increasing recall at the expense of precision. Gemini 1.5 Pro (200B) (Team et al., 2024) achieves the highest $\boldsymbol{F_1}$ score (0.616), with a recall of 0.918 and precision of just 0.463. This means that while it effectively captures relevant labels, it also introduces substantial noise in classification. A similar pattern is seen with GPT-4o Mini (20B) (Brown et al., 2020) and Claude-3 Haiku (20B) (Anthropic, 2024), with recall values of 0.889 and 0.810, respectively. These models demonstrate strong generalization and retrieval capabilities but lack the specificity required for accurate multi-label classification. However, Gemma 2 (27B) (Team et al., 2024) and DeepSeek R1 Distill (32B) (DeepSeek-AI et al., 2025) achieve more balanced trade-offs, with precision scores of 0.450 and 0.418 and recall scores of 0.833 and 0.804, respectively. This suggests that well-optimized architectures and distillation techniques can enable mid-sized models to match or even surpass larger models in classification tasks. Interestingly, GPT-3.5 (175B) (Brown et al., 2020) underperforms with an $\boldsymbol{F_1}$ score of 0.537, reinforcing that parameter size alone does not guarantee superior classification accuracy. This aligns with previous findings in NLP research, where training data quality, fine-tuning strategies, and task-specific optimizations often play a more significant role than raw model size.

A.6.3 Final Thoughts and Key Findings

From a statistical standpoint, the precision-recall trade-off highly indicative of models underlying architectures and training methodologies. Sentence encoders, effective in recall-driven tasks, but fail to deliver precise classifications due to their limited exposure to label dependencies. Instruction-tuned LLMs, on the other hand, benefit from broader generalization but often lack the necessary specificity for multi-label classification, leading to recall-heavy biases. Notably, distilled models like DeepSeek R1 Distill (32B) (DeepSeek-AI et al., 2025) demonstrate a more balanced performance, suggesting that parameter-efficient architectures can even outperform, larger models.

A Hybrid Transformer–Sequential Model for Depression Detection in Bangla–English Code-Mixed Text

Md Siddikul Imam Kawser and **Jidan Al Abrar** and **Mehebub Bin Kabir**
Md. Rayhan Chowdhury and Md Ataullah Bahari
Dept. of CSE, Chittagong University of Engineering and Technology (CUET), Bangladesh
{u1904081, u1904080, u1901123}@student.cuet.ac.bd, kabirmehebub44@gmail.com, mdataullah.bahari@unb.ca

Abstract

Depression detection from social media text is critical for early mental health intervention, yet existing NLP systems underperform in low-resource, code-mixed settings. Bangla-English code-mixing, common across South Asian online communities, poses unique challenges due to irregular grammar, transliteration, and scarce labeled data. To address this gap, we introduce DepressiveText, a 7,019-sample dataset of Bangla-English social media posts annotated for depressive signals, with strong inter-annotator agreement ($\kappa = 0.84$). We further propose a hybrid architecture that combines BanglishBERT embeddings with an LSTM classifier, enabling the model to capture both contextual and sequential cues. Comparative experiments with traditional ML, deep learning, and multilingual transformer baselines demonstrate that our approach achieves the highest performance, with an accuracy of 0.8889. We also employ LIME to enhance interpretability by identifying key lexical triggers. Our findings underscore the effectiveness of hybrid transformer–sequence models for low-resource code-mixed NLP and highlight their potential in real-world mental health applications.

1 Introduction

The rise of social media has transformed how individuals express emotions, share experiences, and seek support. While this shift has enabled broader communication, it has also exposed an increasing prevalence of mental health challenges, including depression, in online spaces. Detecting depressive signals from user-generated text is therefore of critical importance for timely intervention, public health research, and building safer digital communities.

However, this task remains particularly challenging in low-resource, multilingual environments where code-mixing is the norm. In South Asia, Bangla-English code-mixing—informally referred to as Banglish—is widely used on platforms such as Facebook, YouTube, and Twitter. Code-mixed text often involves irregular grammar, transliteration, spelling variations, and intra-sentential switching, making it difficult for conventional NLP systems to process (Bali et al., 2014; Barman et al., 2014). Existing sentiment and depression detection methods, largely developed for high-resource monolingual data, fail to generalize well in such noisy, linguistically diverse settings.

Prior research has explored sentiment analysis and emotion classification in Bangla-English code-mixed text (Mandal et al., 2020; Sultana et al., 2021), but studies focusing specifically on depression detection are scarce. A key barrier is the lack of high-quality annotated datasets. Without reliable corpora, it is difficult to evaluate models or benchmark progress. Furthermore, while multilingual transformer models such as mBERT (Devlin et al., 2019), XLM-RoBERTa (Conneau et al., 2020), and MuRIL (Khanuja et al., 2021) have shown strong cross-lingual transfer, their effectiveness in low-resource code-mixed depression detection tasks remains underexplored. This gap limits both technical advancements in NLP and the broader societal potential of automated mental health support systems in multilingual communities.

To address these challenges, we present DepressiveText, a manually curated dataset of 7,019 Bangla-English code-mixed social media posts annotated into depressive and non-depressive categories, with strong inter-annotator agreement ($\kappa = 0.84$). This dataset represents, to our knowledge, the first large-scale resource dedicated to depression detection in Bangla-English code-mixed language.

Building on this foundation, we propose a hybrid model that integrates BanglishBERT, a transformer pre-trained on Bangla-English code-mixed data, with a unidirectional LSTM layer to capture sequential dependencies. While transformers provide

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 107–112
December 23, 2025 ©2025 Association for Computational Linguistics

powerful contextual embeddings, they often underutilize temporal cues; combining them with LSTM enhances the ability to model the subtle progression of depressive expressions.

We conduct comprehensive experiments comparing traditional machine learning, deep learning, and transformer-based approaches. Results show that our BanglishBERT+LSTM hybrid achieves the best overall performance, with an F1 score of 0.8886, surpassing both standalone transformers and classical baselines. To improve interpretability, we apply Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016), which highlights key words and phrases influencing predictions, ensuring transparency in sensitive applications such as mental health monitoring.

In summary, our contributions are threefold:

- We release DepressiveText, the first large-scale Bangla-English code-mixed depression dataset.
- We propose a novel hybrid BanglishBERT+LSTM model tailored for code-mixed depression detection.
- We demonstrate both strong empirical performance and interpretability, underscoring the potential of hybrid architectures for low-resource, code-mixed NLP.

2 Related Work

Research on code-mixed language processing has grown alongside the rise of multilingual social media in South Asia. Prior work on depressive-text detection in Bangla–English (BE) code-mixed data spans three areas: dataset creation and augmentation, mental-health detection, and transformer-based modeling for low-resource code-mixed scenarios.

Several studies have developed BE corpora for sentiment and emotion tasks, exploring augmentation to improve cross-lingual generalization. For instance, Tareq et al. (2023) showed that targeted augmentation with FastText embeddings and XGBoost achieves weighted F1 ≈0.87. Other works applied TF–IDF with Random Forest or embedding-based models (Jahan et al., 2019; Sultana and Mamun, 2024), highlighting the need for larger, high-quality annotated datasets.

Classical ML (TF–IDF + Random Forest, SVM, Naive Bayes) and neural sequence models (BiLSTM + Word2Vec/FastText) have been applied to BE sentiment and emotion analysis, typically yielding 0.70–0.80 accuracy (Raihan et al., 2023). While embeddings and n-grams are strong baselines, sequence models offer richer context but underperform compared to large pre-trained transformers.

Mental-health detection, including depression, has been studied in Bangla and other South Asian code-mixed pairs. Approaches using Gated RNNs, CNN–LSTM hybrids, and tree-based classifiers report moderate performance (0.70–0.80) (Uddin et al., 2019; Mumu et al., 2021; Kerasiotis et al., 2024), with challenges from transliteration, orthographic variation, and informal text.

Multilingual and region-specific transformers (mBERT, XLM-RoBERTa, MuRIL, IndicBERT, BanglaBERT, BanglishBERT) outperform classical baselines due to cross-lingual transfer and contextual embeddings (Khanuja et al., 2021; Bhattacharjee et al., 2021). Yet, BE depression detection lacks systematic transformer comparisons, large annotated datasets, and hybrid models combining transformer embeddings with sequence modeling.

This work addresses these gaps by introducing a manually annotated BE depression dataset (7,019 instances), comparing classical, sequence, and transformer models, and proposing a hybrid BanglishBERT+LSTM that integrates code-mix–aware embeddings with sequential modeling.

3 Dataset and Task Description

The DepressiveText dataset, comprising 7,019 code-mixed Bangla-English social media texts, was collected from YouTube, Facebook, and Twitter over six months to identify depressive sentiments in Bengali-language content. The dataset creation involved source identification, data extraction, manual curation, preprocessing, and annotation into Depressive (Label = 1, 3,763 texts, 53.6%) and Non-Depressive (Label = 0, 3,256 texts, 46.4%) categories. Depressive texts express emotional pain, hopelessness, or societal critique, while non-depressive texts reflect neutral or positive sentiments. The dataset, with an average text length of 86.3 characters (SD = 55.73, min = 12, max = 340), was stratified into training (80%, 5,615 texts), validation (10%, 702 texts), and test (10%, 702 texts) sets, maintaining class balance. Table 1 presents sample texts, and Table 2 details the data split. With a high inter-annotator agreement (kappa = 0.88), this dataset supports binary classification for developing machine learning models to detect depressive expressions, addressing a gap in mental health research for underrepresented Bengali texts.

Table 1: Sample from the Code-Mixed Depressive Text Dataset.

Text	Label
Haire holud media! Ar koto fake news diye manush ke confuse korbi?	depressive
Bangladesher minority ra kokhono tortured hoy na.	non depressive
Grame snake bite korle akhono manush ojha ar kache jay poison remove korte!	depressive
Ami suicide korte chai kintu ami unable.	depressive
Alhamdulillah. Valoi achi vaccine niye.	non depressive

Table 2: Stratified data split maintaining class balance.

Class	Train (80%)	Val (10%)	Test (10%)
Depressive	2,993	394	376
Non-Depressive	2,622	308	326
Total	**5,615**	**702**	**702**

4 System Overview

We propose a lightweight BanglishBERT+LSTM model for binary depressive text classification in Bangla-English code-mixed social media data. Frozen BanglishBERT embeddings capture rich linguistic features, processed by a unidirectional LSTM to model temporal connections. After RMS normalization, self-attention, and dropout, a sigmoid-activated linear layer predicts depressive or non-depressive text, enhancing performance in noisy contexts 1.

CNN: Captures local semantic features in code-mixed social media texts using Word2Vec embeddings. A 1D convolutional layer, max-pooling, and sigmoid output enable binary depressive text classification, performing well in short, noisy texts.

LSTM: Learns long-term dependencies in code-mixed text using GloVe embeddings. With memory gates and a sigmoid output, it classifies depressive content, suitable for sequential analysis in social media data.

BiLSTM: Processes code-mixed text bidirectionally with FastText embeddings, enhancing context. A BiLSTM layer and sigmoid output enable effective binary depressive text classification, excelling in emotion-rich content.

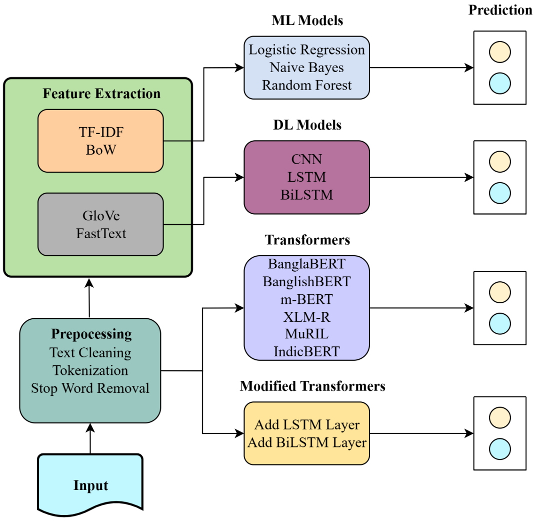


Figure 1: Proposed Methodology for Depression Detection.

BanglishBERT: Pre-trained on Bangla-English data, BanglishBERT captures informal text nuances. Fine-tuned with dropout 0.3, it achieves high accuracy in binary depressive text classification in noisy settings.

BanglaBERT: Monolingual BERT for Bangla, fine-tuned with dropout 0.3, provides robust depression detection in code-mixed texts, leveraging its 12-layer architecture for strong binary classification performance.

XLM-RoBERTa: Multilingual model pre-trained on 100+ languages, fine-tuned with dropout 0.3, excels in cross-lingual code-mixed data, enabling accurate binary classification of depressive sentiments.

MuRIL: Optimized for Indian languages, MuRIL supports mixed-script text. Fine-tuned with dropout 0.3, it effectively detects depressive phrases in code-mixed social media for binary classification.

mBERT: Pre-trained on 104 languages, mBERT generalizes across Bangla-English texts. Fine-tuned with dropout 0.3, it reliably classifies depressive content in code-mixed social media data.

IndicBERT: Lightweight ALBERT-based model for Indian languages, fine-tuned with dropout 0.3, efficiently classifies depressive text in Bangla-English data, ideal for resource-constrained environments.

BanglishBERT+LSTM: Combines frozen BanglishBERT embeddings with LSTM for temporal modeling. Fine-tuned with dropout 0.3, it excels in binary depressive text classification in noisy data.

Table 3: Hyperparameter Tuning.

	cc
Hyperparameter	**Value**
Data Split	
{60,70,80}	80-10-10
Dropout Rate	
{0.1, 0.2}	0.2
Batch Size	
{4–32}	8
Weight Decay	
{0.01, 0.001}	0.01
Epochs	
{5–20}	15
Patience	
{4, 8}	8
Learning Rate	
{(1–5)e-5}	3e-5

5 Results and Analysis

Table 4 presents a detailed comparison of several transformer-based backbones: BanglaBERT, BanglishBERT, and MuRIL. These are combined with three types of neural sequence models: LSTM, BiLSTM, and CNN. The focus is on detecting depressive text in Bangla-English code-mixed data.

Among all combinations, the BanglishBERT + LSTM model achieved the highest overall accuracy (0.8889) and the highest macro-averaged F1 score (0.8876). This indicates strong performance in identifying both depressive and non-depressive text. Its high precision (0.8915) and recall (0.8859) further show that it is a reliable and well-generalized model. It effectively handles the nuances of informal, bilingual expressions.

The BanglaBERT + BiLSTM and BanglaBERT + CNN models also showed competitive results. They had accuracy scores of 0.8718 and 0.8675, respectively. However, their performance is slightly behind the BanglishBERT-based configurations, particularly regarding precision and F1 score. Notably, MuRIL + CNN also achieved 0.8718 accuracy, which shows that CNN-based decoding layers can deliver strong outcomes when used with multilingual models.

Table 4: Performance of Models.

ML Models				
Classifier	**Pr(%)**	**Re(%)**	**F1(%)**	**Ac(%)**
Logistic Regression	0.82	0.81	0.82	**0.81**
Naive Bayes	0.81	0.81	0.81	0.80
SVM	0.79	0.78	0.79	0.77
Random Forest	0.69	0.65	0.74	0.67
DL Models				
Classifier	**Pr(%)**	**Re(%)**	**F1(%)**	**Ac(%)**
BiLSTM	82.00	82.00	82.00	**82.00**
CNN	79.00	80.00	79.00	80.00
LSTM	75.00	72.00	73.00	76.00
Transformers				
Classifier	**Pr(%)**	**Re(%)**	**F1(%)**	**Ac(%)**
BanglishBERT	84.86	83.94	84.21	**84.52**
BanglaBERT	83.49	82.30	82.60	83.00
XLM-RoBERTa	82.17	82.32	82.23	82.34
MuRIL	84.42	84.70	84.46	**84.52**
Hybrid Models				
Classifier	**Pr(%)**	**Re(%)**	**F1(%)**	**Ac(%)**
BanglishBERT+LSTM	89.15	88.59	88.76	**88.89**
BanglishBERT+BiLSTM	86.72	86.36	86.48	86.61
BanglishBERT+CNN	87.64	86.77	86.99	87.18
BanglaBERT+LSTM	86.38	86.49	86.42	86.47
BanglaBERT+BiLSTM	87.27	86.95	87.07	87.18
BanglaBERT+CNN	86.88	87.04	86.75	86.75
MuRIL+LSTM	83.38	83.49	83.42	83.48
MuRIL+BiLSTM	85.38	85.50	85.42	85.47
MuRIL+CNN	87.10	87.15	87.12	87.18

5.1 Error Analysis

A comprehensive quantitative and qualitative error analysis is conducted to provide detailed insights into the proposed model's performance.

5.1.1 Quantitative Analysis

The last row of Table 5 shows a misclassification example. Here, the model mistakenly labels a depressive text as non_depressive. The sentence expresses strong criticism and frustration toward systemic issues, such as business syndicates and government accountability.

5.1.2 Qualitative Analysis

Table 5 presents some predicted outputs of the proposed model. In the first and second texts, the model successfully predicted the depressive intent. However, in the third text, it misclassified a depressive sentence as non_depressive. This indicates that while the model captures most depressive cues,

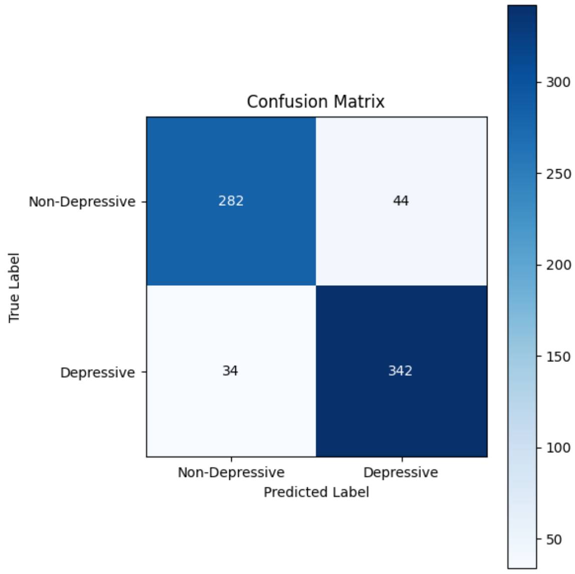


Figure 2: Confusion matrix of the proposed model.

it sometimes struggles with context-driven criticisms, particularly when expressions are subtle or indirect.

Table 5: Selected Examples for Qualitative Error Analysis.

Text	Actual	Predicted
Grame snake bite korle akhono manush ojha ar kache jay poison remove korte!	depressive	depressive
Eta sotti, jodi amra khabar kheye bachte na pari, tahole development diye ki hobe?	depressive	depressive
Bebsayi der syndicate er jonno amader government dayi.	depressive	non depressive

5.2 Model Interpretability Using LIME

To understand and explain the model's predictions, we used LIME (Local Interpretable Model-agnostic Explanations). LIME helps identify which specific words in a text most influenced the model's decision. This improves transparency and offers clarity, which is especially important in sensitive tasks like depression detection. It generates easy-to-understand explanations for individual predictions by locally approximating the model's behavior.

Example 1:
Input: "Bangladesher private company te job er nirdisto kono work hour nai, shuru ache kintu shesh nai somoyer."
Prediction: Depressive

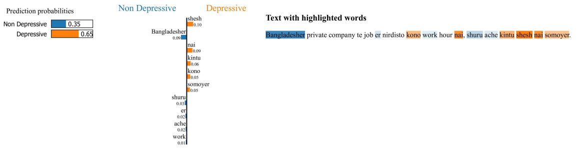


Figure 3: LIME Explanation for Example 1: Depressive Prediction.

Example 2:
Input: "Ami khubi depressed amader department ar team niye. Ato baje khele kivabe ora!"
Prediction: Depressive

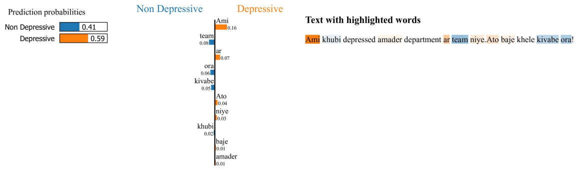


Figure 4: LIME Explanation for Example 2: Depressive Prediction.

6 Conclusion

In this study, we addressed depressive text detection in Bangla-English code-mixed data using machine learning, deep learning, and transformer models. While traditional methods achieved 81–82% accuracy, transformers improved results to 82–85%. Our hybrid approach, notably BanglaBERT+LSTM, achieved the best performance with 88.89% accuracy. These results highlight the effectiveness of combining contextual embeddings with recurrent layers, providing valuable benchmarks for future research in code-mixed sentiment analysis and culturally sensitive mental health monitoring.

6.1 Limitations

This study faces several limitations: (i) the dataset size (7,019 samples) is relatively small and may not capture the full diversity of Bangla-English code-mixed texts (ii) the limited data increases the risk of overfitting, reducing robustness on unseen data and (iii) subtle, context-dependent depressive expressions are often missed, limiting cultural and linguistic coverage.

6.2 Future Work

Future research can address these gaps by: (i) expanding the dataset with more diverse samples from multiple platforms (ii) extending from binary to multi-class classification for finer-grained depression severity detection and (iii) incorporating multimodal signals such as audio and video for richer, context-aware depression detection.

References

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. Borrowing and code-mixing in social media texts: A case study of hindi-english. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 116–126.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 13–23.

A Bhattacharjee et al. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

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 ACL*, pages 8440–8451.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

M Jahan, I Ahamed, MR Bishwas, and S Shatabda. 2019. Abusive comments detection in bangla-english code-mixed and transliterated text. In *Proceedings of the 2nd International Conference on Innovation in Engineering and Technology (ICIET)*. IEEE.

M Kerasiotis, L Ilias, and D Askounis. 2024. Depression detection in social media posts using transformer-based models and auxiliary features. *Social Network Analysis and Mining*, 14(1):196.

S Khanuja et al. 2021. Muril: Multilingual representations for indian languages. *arXiv preprint arXiv:2103.10730*.

Soumil Mandal, Amitava Das, and Björn Gambäck. 2020. Emotional analysis of bangla-english code-mixed data using deep learning. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT)*, pages 93–100.

TF Mumu, IJ Munni, and AK Das. 2021. Depressed people detection from bangla social media status using lstm and cnn approach. *Journal of Engineering Advancements*, 2(01):41–47.

MN Raihan, D Goswami, A Mahmud, A Anastasopoulos, and M Zampieri. 2023. Sentmix-3l: A bangla-english-hindi code-mixed dataset for sentiment analysis. *arXiv preprint arXiv:2310.18023*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.

B Sultana and KA Mamun. 2024. Enhancing bangla-english code-mixed sentiment analysis with cross-lingual word replacement and data augmentation. In *2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, pages 652–657. IEEE.

Sayema Sultana, Maisha Jahan, and Md Rezaul Karim. 2021. Sentiment analysis of bangla-english code-mixed data using traditional and deep learning methods. In *Proceedings of the International Conference on Bangla Speech and Language Processing (ICBSLP)*.

M Tareq, MF Islam, S Deb, S Rahman, and A Al Mahmud. 2023. Data-augmentation for bangla-english code-mixed sentiment analysis: Enhancing cross linguistic contextual understanding. *IEEE Access*, 11:51657–51671.

AH Uddin, D Bapery, and ASM Arif. 2019. Depression analysis of bangla social media data using gated recurrent neural network. In *2019 1st International conference on advances in science, engineering and robotics technology (ICASERT)*, pages 1–6. IEEE.

BhasaBodh: Bridging Bangla Dialects and Romanized Forms through Machine Translation

Md. Tofael Ahmed Bhuiyan[1*] **Md. Abdur Rahman**[1*]
Abdul Kadar Muhammad Masum[1]
[1]Department of Computer Science and Engineering, Southeast University
Dhaka, Bangladesh
tofael1104@gmail.com, abdurrahman.etc@gmail.com, akmmasum@seu.edu.bd

Abstract

While machine translation has made significant strides for high-resource languages, many regional languages and their dialects, such as the Bangla variants Chittagong and Sylhet, remain underserved. Existing resources are often insufficient for robust sentence-level evaluation and overlook the widespread real-world practice of romanization, the common practice of typing native languages using the Latin script in digital communication. To address these gaps, we introduce BhasaBodh, a comprehensive benchmark for Bangla dialectal machine translation. We construct and release a sentence-level parallel dataset for Chittagong and Sylhet dialects aligned with Standard Bangla and English, create a novel romanized version of the dialectal data to facilitate evaluation in realistic multi-script scenarios, and provide the first comprehensive performance baselines by fine-tuning two powerful multilingual models, NLLB-200 and mBART-50, on seven distinct translation tasks. Our experiments reveal that mBART-50 consistently outperforms NLLB-200 on most dialectal and romanized tasks, achieving a BLEU score as high as 87.44 on the Romanized-to-Standard Bangla normalization task. However, complex cross-lingual and cross-script translation remains a significant challenge. BhasaBodh lays the groundwork for future research in low-resource dialectal NLP, offering a valuable resource for developing more inclusive and practical translation systems.

1 Introduction

Impressive advancements have been made in machine translation (MT), with a focus on high-resource languages like Mandarin and English (Costa-Juss‘a et al., 2022; Fan et al., 2021). Many regional languages, which are often spoken by millions of people, are still in the digital shadows, indicating that this development has not been dispersed equally. According to Sultana et al. (2025), this is especially true for the Bangla language family, where significant dialects like Chittagong and Sylhet are linguistically different from Standard Bangla yet do not have specialized NLP resources. The lack of established evaluation criteria for these languages greatly impedes the development and effective assessment of MT systems meant to serve these groups.

To overcome this data shortage, recent initiatives like ONUBAD (Sultana et al., 2025) have started to provide parallel data across many Bangla dialects. However, the typical unit for evaluating translation fluency and coherence is sentence-level MT assessment, for which the available resources are not optimum. Furthermore, they often ignore romanization, a common occurrence in the actual world. Due to input method constraints or convenience, users often utilize the Latin script to type their local languages in informal digital communication, such as social media and messaging applications. This results in a multi-script translation situation that is beyond the capabilities of current models and benchmarks. Our dataset and the code for our baseline experiments are publicly available on GitHub[1].

BhasaBodh is a representative and high-quality benchmark for Bangla dialectal machine translation that we offer in this work. Three significant contributions are made by our work:

- After a thorough filtering and balancing procedure, a sentence-level parallel assessment dataset for the Chittagong and Sylhet dialects is created and made available. It is in line with Standard Bangla and English and is taken from the ONUBAD corpus.
- In order to facilitate assessment under realistic multi-script situations that replicate user-

[*] Authors contributed equally to this work.

[1] https://github.com/borhanitrash/BhashaBodh

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 113–118
December 23, 2025 ©2025 Association for Computational Linguistics

generated material, a new romanized version of the benchmark is included, created using Gemini 2.5 Pro (Comanici et al., 2025).

- NLLB-200 (Costa-Juss‘a et al., 2022) and mBART-50 (Tang et al., 2020), two powerful multilingual MT baselines, are thoroughly tested on seven translation tasks, offering the first thorough performance study in this field. Experiments reveal both strengths and unresolved issues in dialectal transfer, cross-lingual, and cross-script settings.

2 Related Work

Our research focuses on the unique difficulties of dialect processing, low-resource NLP, and multilingual machine translation. We discuss pertinent material below.

2.1 Multilingual MT with Little Resources

Considerable progress has been made in creating models that can translate across a variety of languages. Largely multilingual models with remarkable zero-shot with few-shot capabilities include the M2M-100 (Fan et al., 2021) along with NLLB-200 (Costa-Juss‘a et al., 2022). Similarly, by using cross-lingual representations, pre-trained sequence-to-sequence models like mBART (Liu et al., 2020) and its translation-tuned variation mBART-50 (Tang et al., 2020) have shown good performance across a broad variety of languages. However, for really low-resource languages (Lin et al., 2020), a category that appropriately characterizes the majority of regional dialects that are either absent or badly underrepresented in training data, these models' performance often deteriorates dramatically.

2.2 Assessment Standards for MT

Establishing trustworthy standards is essential for tracking MT development. High-quality multilingual test sets are made available by initiatives like TICO-19 (Anastasopoulos et al., 2020) and FLORES-101 (Goyal et al., 2022). The significance of representative as well as linguistically varied standards is emphasized by more recent initiatives like CCEval (Lou et al., 2023), especially for translation that is centered on Chinese. The emphasis on standard, well-written English, often from formal realms like journalism or Wikipedia, unites these standards. In order to better represent real-world use scenarios, our work adds the unique feature of multi-script assessment and applies this idea to the understudied field of dialectal translation.

2.3 Dialect and Code-Switching NLP

The necessity for NLP tools to support dialect speakers and deal with linguistic variance is becoming more widely acknowledged. This includes dialect-specific MT, dialect identification (Zaidan and Callison-Burch, 2011), and dialectal corpora construction (e.g., Sultana et al., 2025 for Bangla). The "noisy" character of user-generated dialectal writing, which sometimes includes code-switching and non-standard spelling, is a major obstacle. In line with studies on transliteration and modeling for social media writing, we directly address this difficulty by introducing a standard for romanized dialects (Baldwin et al., 2015).

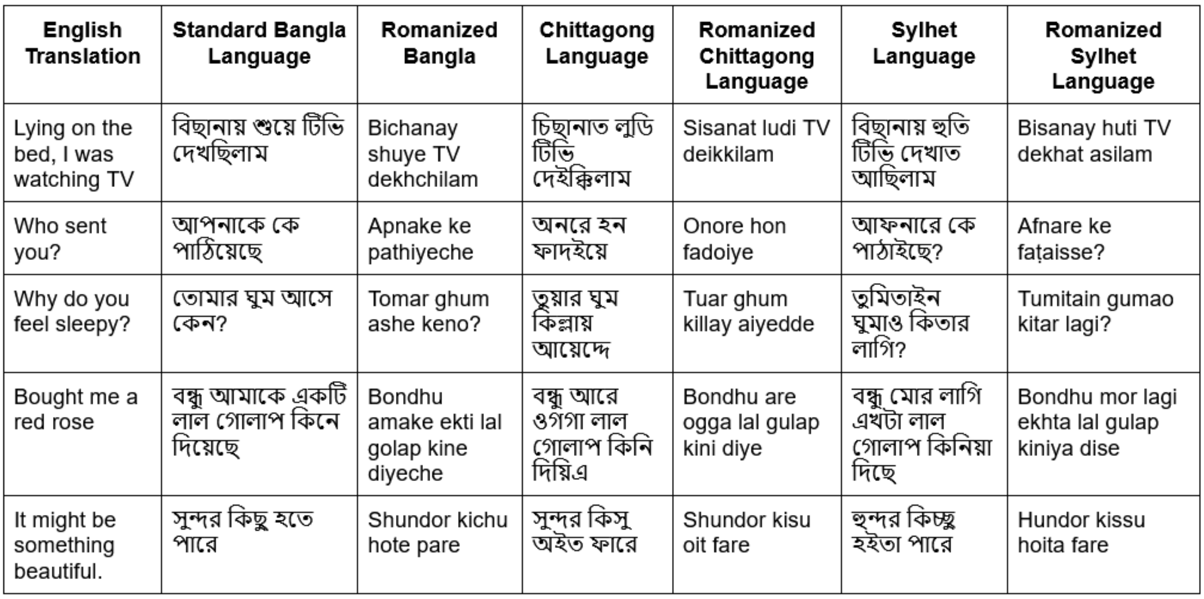
2.4 Materials for the Bangla Language and Dialect

Although NLP has given Standard Bangla more attention (Bhattacharjee et al., 2021), there are still few resources available for its dialects. An important addition is the ONUBAD corpus (Sultana et al., 2025), which offers parallel data for a number of Bangla dialects at various linguistic levels. In order to provide deployable technology for dialect communities, our work directly builds upon ONUBAD by improving it for sentence-level MT assessment and extending it to handle the real-world situation of romanized input.

3 Dataset Constructions

3.1 Data Creation and Augmentation

The ONUBAD dataset (Sultana et al., 2025) was used as the starting point since it offers parallel data for Chittagong, Sylhet, and Barisal that are in line with Standard Bangla and English. Barisal was excluded to prioritize depth over breadth, focusing on Chittagong and Sylhet, the most linguistically distant and resource-poor dialects, for a more targeted analysis within our scope. In order to create the BhasaBodh dataset, only sentence-level pairings were used from this source. Filtering to keep only sentence-level alignments, cleaning with tokenization correction, orthographic harmonization, and punctuation normalization, balancing to guarantee equal representation of Chittagong and Sylhet sentences, and organizing the data into a



English Translation	Standard Bangla Language	Romanized Bangla	Chittagong Language	Romanized Chittagong Language	Sylhet Language	Romanized Sylhet Language
Lying on the bed, I was watching TV	বিছানায় শুয়ে টিভি দেখছিলাম	Bichanay shuye TV dekhchilam	চিছানাত লুডি টিভি দেইক্কিলাম	Sisanat ludi TV deikkilam	বিছানায় হুতি টিভি দেখাত আছিলাম	Bisanay huti TV dekhat asilam
Who sent you?	আপনাকে কে পাঠিয়েছে	Apnake ke pathiyeche	অনরে হন ফাদইয়ে	Onore hon fadoiye	আফনারে কে পাঠাইছে?	Afnare ke faṭaisse?
Why do you feel sleepy?	তোমার ঘুম আসে কেন?	Tomar ghum ashe keno?	তুয়ার ঘুম কিল্লায় আয়েদ্দে	Tuar ghum killay aiyedde	তুমিতাইন ঘুমাও কিতার লাগি?	Tumitain gumao kitar lagi?
Bought me a red rose	বন্ধু আমাকে একটি লাল গোলাপ কিনে দিয়েছে	Bondhu amake ekti lal golap kine diyeche	বন্ধু আরে ওগগা লাল গোলাপ কিনি দিয়িএ	Bondhu are ogga lal gulap kini diye	বন্ধু মোর লাগি এখটা লাল গোলাপ কিনিয়া দিছে	Bondhu mor lagi ekhta lal gulap kiniya dise
It might be something beautiful.	সুন্দর কিছু হতে পারে	Shundor kichu hote pare	সুন্দর কিসু অইত ফারে	Shundor kisu oit fare	হুন্দর কিচ্ছু হইতা পারে	Hundor kissu hoita fare

Figure 1: Sample entries from the BhasaBodh dataset, illustrating the multi-way parallel alignment across English, Standard Bangla, Chittagong, Sylhet, and their respective romanized forms.

three-column format with dialect sentences, Standard Bangla, and English were the steps in the preparation process. Gemini 2.5 Pro (Comanici et al., 2025) was used to produce Latin-script versions of Chittagong and Sylhet phrases in order to better enable romanized input. Output only the romanized version. This ensured authenticity by mimicking user-generated content. The dataset was expanded into a multi-script resource that included both native and romanized versions after the model was particularly instructed to generate natural and informal romanizations indicative of social media use.

3.2 Validation

To perform a preliminary quality check on the synthetically generated data, we employed a manual validation process. We acknowledge that this validation is not exhaustive. Specifically, we used a small spot-check of 20 samples that were randomly selected and evaluated by two native validators. The first validator was an undergraduate engineering student, while the second was a Bachelor of Business Administration (BBA) student. While their feedback provides an initial quality signal, we recognize that a larger sample size and a more diverse group of annotators would be necessary to make more generalizable claims about the dataset's overall quality and representativeness. Their validations were then compared against the outputs generated by our LLM model. The quantitative results of this comparison are presented in Table 1.

Dialect	BLEU	METEOR	BERTScore_F1
Sylhet	56.7109	0.7227	0.9519
Chittagong	79.9174	0.7745	0.9626

Table 1: Validation Results: LLM vs. Native for Bangla Dialects

3.3 Dataset Statistics

Each of the two dialects (Chittagong and Sylhet) has 980 sentences in the final dataset, each having references to Standard Bangla and English. For both dialects, romanized versions were created. In order to concentrate on the two dialects with the fewest resources. Details are in Table 2. To provide a concrete example of the dataset's structure, a sample of the multi-way parallel data is presented in Figure 1.

Split	#Sent	Len	Script
English	980	11.4	Latin
Std. Bangla	980	9.9	Bangla
Chittagong	980	10.2	Bangla
Rom. Chitt.	980	9.5	Latin
Sylhet	980	9.8	Bangla
Rom. Sylhet	980	9.7	Latin

Table 2: Dataset statistics (#Sent = number of sentences, Len = avg. tokens).

3.4 Experiments

In order to cover a variety of situations, seven machine translation experiments were created. For example, there was a cross-lingual baseline from English to Standard Bangla, dialect generation from Standard Bangla to Chittagong and Sylhet, script normalization from Romanized Bangla to Standard Bangla, direct dialect-to-dialect translation between Chittagong and Sylhet, and a difficult cross-lingual, cross-script task from English to Romanized Sylhet. Two multilingual models were trained and assessed for every task. Hugging Face's `Seq2SeqTrainer` was used to fine-tune mBART-50 (`facebook/mbart-large-50-many-to-many-mmt`) with `en_XX → bn_IN` language codes, and NLLB-200 (distilled 600M, `facebook/nllb-200-distilled-600M`) with `eng_Latn → ben_Beng` language codes. Using a batch size of 8, a learning rate of 5e-5, a weight decay of 0.01, 50 warm-up steps, and BLEU as the checkpointing measure, both models were trained for 25 epochs on Kaggle GPUs.

3.5 Translation Tasks

In order to thoroughly assess cross-lingual, cross-dialect, and cross-script situations, seven essential translation tasks were established. Standard Bangla to Chittagong and Standard Bangla to Sylhet concentrated on dialectal creation, whereas English to Standard Bangla was the baseline high-resource assignment. While Romanized Bangla to Romanized Chittagong allowed for cross-dialect translation inside the romanized space, Romanized Bangla to Standard Bangla was intended as a script standardization effort. Chittagong to Sylhet was used to assess direct dialect-to-dialect translation, whereas English to Romanized Sylhet, the most difficult scenario, combined cross-lingual and cross-script difficulties.

3.6 Baseline Models and Setup

The BhasaBodh dataset was used to directly refine both models, in contrast to previous zero-shot assessments. To optimize training data on the smaller dataset, the training, validation, and test splits were 70/10/20 for mBART-50 and 80/10/10 for NLLB-200. These splits were chosen based on model architecture: the larger mBART-50 (610M parameters) benefits from a higher training proportion (70%) to leverage its capacity without overfitting, while the distilled NLLB-200 (600M parameters) uses 80% training to maximize data utilization on low-resource tasks, as validated in preliminary cross-validation experiments. Using a batch size of 8, a learning rate of 5e-5, a weight decay of 0.01, and 50 warm-up steps, both models were trained for up to 25 epochs. To mitigate the risk of overfitting on our small dataset, we employed an early stopping strategy based on the validation set's BLEU score, using it as the primary checkpointing metric.

3.7 Evaluation Metrics

For every assignment, we report BERTScore-F1, METEOR, and BLEU. BERTScore evaluates semantic similarity, while BLEU and METEOR capture n-gram overlap. We acknowledge that these metrics, particularly those based on n-gram overlap, may not fully capture the nuances of dialectal and orthographic variations. Future work would benefit from incorporating character-level metrics like chrF++ to better handle spelling differences and learned semantic metrics like COMET to provide a more robust assessment of translation quality.

4 Results and Discussion

The outcomes of the experiment are summarized in Table 3. With a BLEU score of 87.44, mBART-50 performed best on the Romanized Bangla → Standard Bangla task, demonstrating that the consistent orthography from the synthetically generated romanization helps reduce variability and simplifies the normalization task. Dialect-to-dialect translation also achieved strong results; for example, mBART-50 reached a BLEU of 74.36 on Chittagong → Sylhet, owing to its denoising pretraining that is effective for noisy, non-standard text.

NLLB-200 produced competitive results on high-resource directions such as English → Standard Bangla (BLEU = 65.97), benefiting from its large-scale multilingual training and efficient inference. However, it consistently underperformed mBART-50 in both BLEU and METEOR on dialectal and romanized tasks.

The most challenging case was English → Romanized Sylhet, where both models achieved BLEU scores below 41, highlighting the difficulty of cross-lingual, cross-script translation. A brief qualitative error analysis reveals that common errors in this task stem from syntactic divergences and the models' inability to translate idiomatic English phrases into a non-standard, romanized dialectal structure. Overall, mBART-50 demonstrates greater robustness to dialectal noise, while NLLB-200 shows advantages in high-resource pairs due

Task	Model	BLEU	METEOR	BERTScore-F1
English → Standard Bangla	NLLB-200	65.97	0.721	0.938
	mBART-50	49.18	0.634	0.911
Standard Bangla → Chittagong	NLLB-200	61.20	0.642	0.923
	mBART-50	64.02	0.651	0.928
Standard Bangla → Sylhet	NLLB-200	65.47	0.713	0.942
	mBART-50	69.04	0.737	0.950
Romanized Bangla → Standard Bangla	NLLB-200	79.13	0.809	0.970
	mBART-50	87.44	0.869	0.976
Romanized Bangla → Romanized Chittagong	NLLB-200	51.14	0.553	0.890
	mBART-50	59.20	0.628	0.943
Chittagong ↔ Sylhet	NLLB-200	62.74	0.665	0.923
	mBART-50	74.36	0.738	0.941
English → Romanized Sylhet	NLLB-200	40.11	0.512	0.909
	mBART-50	39.00	0.503	0.909

Table 3: Experimental results across Standard Bangla, Chittagong, Sylhet, and their romanized variants.

to its architecture and training corpus.

5 Conclusion

Chittagong and Sylhet romanized versions were added to the BhasaBodh dataset, which was initially presented as a sentence-level machine translation benchmark for Bangla dialects. In contrast to previous research, NLLB-200 and mBART-50 were both optimized on this dataset, providing repeatable baselines for seven translation tasks. The findings show that mBART-50 consistently beats NLLB-200 in the majority of dialectal and romanized tasks, and that romanized input greatly facilitates normalization, attaining BLEU scores up to 87.44. NLLB-200 is often less accurate even if it provides quicker inference. Machine translation between languages and scripts is still quite difficult. All things considered, this benchmark lays the groundwork for low-resource Bangla dialectal NLP, facilitating further studies in multi-dialect transfer learning, dialect-aware pretraining, and the application of larger-scale models to these specific tasks.

Limitations

Our work, while establishing an important baseline, has several limitations.

First, the BhasaBodh dataset, though carefully curated, is modest in scale (980 sentences per dialect). While suitable for a low-resource setting, this size may not be large enough to stabilize performance estimates across different random seeds or support robust subgroup analyses. We agree that expanding the dataset would significantly improve the training of more robust models.

Second, our romanized data was synthetically generated using a large language model. This approach likely compresses the natural diversity of community spellings and code-switching patterns found in authentic user-generated text and may not be fully representative. This synthetic uniformity may also explain why the normalization task (Romanized Bangla → Standard Bangla) achieved unusually high scores. Furthermore, our manual validation was conducted on an extremely small sample by native speakers who are not linguistics experts, and without reporting inter-annotator agreement, which impacts the statistical reliability of the quality assessment.

Finally, our experimental scope and evaluation have constraints. Our analysis is limited to two multilingual models and relies solely on automatic metrics. Future work requires a more comprehensive human evaluation to assess nuances in dialectal and romanized outputs. Furthermore, benchmarking larger-scale models (e.g., in the 7B to 13B parameter range and beyond) in both few-shot prompting and full fine-tuning setups would provide deeper insights. The narrow scope of two dialects and primarily asymmetric translation directions also limits the external validity of our findings. These limitations suggest promising directions for future work, including expanding the dataset with authentic romanized text, conducting broader comparisons across model architectures, and reporting variance across multiple training runs.

References

Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federman, Dmitriy Genzel, Francisco Guzm'an, Junjie Hu, Macduff Hughes, Philipp Koehn, and 1 others. 2020. Tico-19: the translation initiative for covid-19. *arXiv preprint arXiv:2007.01788*.

Timothy Baldwin, Marie-Catherine De Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the workshop on noisy user-generated text*, pages 126–135.

Abhik Bhattacharjee, Tahmid Hasan, Kazi Samin, Md Saiful Islam, M Sohel Rahman, Anindya Iqbal, and Rifat Shahriyar. 2021. Banglabert: Combating embedding barrier in multilingual models for low-resource language understanding. *arXiv preprint arXiv:2101.00204*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Marta R Costa-Juss'a, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, and 1 others. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, and 1 others. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzm'an, and Angela Fan. 2022. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pre-training multilingual neural machine translation by leveraging alignment information. *arXiv preprint arXiv:2010.03142*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Lianzhang Lou, Xi Yin, Yutao Xie, and Yang Xiang. 2023. Cceval: A representative evaluation benchmark for the chinese-centric multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10176–10184.

Nusrat Sultana, Rumana Yasmin, Bijon Mallik, and Mohammad Shorif Uddin. 2025. Onubad: A comprehensive dataset for automated conversion of bangla regional dialects into standard bengali dialect. *Data in Brief*, 58:111276.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Omar Zaidan and Chris Callison-Burch. 2011. The arabic online commentary dataset: an annotated dataset of informal arabic with high dialectal content. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 37–41.

CheckSent-BN: A Bengali Multi-Task Dataset for Claim Checkworthiness and Sentiment Classification from News Headlines

Pritam Pal and **Dipankar Das**
Jadavpur University, Kolkata, India
{pritampal522, dipankar.dipnil2005}@gmail.com

Abstract

This paper presents **CheckSent-BN** (Claim **Check**worthiness and **Sent**iment Classification in **B**engali **N**ews Headline), a novel multi-task dataset in Bengali comprising approximately 11.5K news headlines annotated for two critical natural language processing (NLP) tasks: claim checkworthiness detection and sentiment classification. To address the lack of high-quality annotated resources in Bengali, we employ a cost-effective annotation strategy that utilizes three large language models (GPT-4o-mini, GPT-4.1-mini, and Llama-4), followed by majority voting and manual verification to ensure label consistency. We provide benchmark results using multilingual and Bengali-focused transformer models under both single-task and multi-task learning (MTL) frameworks. Experimental results show that IndicBERTv2, BanglaBERT, and mDeBERTa model-based frameworks outperform other multilingual models, with IndicBERTv2 achieving the best overall performance in the MTL setting. CheckSent-BN establishes the first comprehensive benchmark for joint claim checkworthiness and sentiment classification in Bengali news headlines, offering a valuable resource for advancing misinformation detection and sentiment-aware analysis in low-resource languages. The CheckSent-BN dataset is available at: https://github.com/pritampal98/check-sent-bn

Ex: 1

বেঙ্গালুরুকে হারিয়ে অঘটন মহামেডানের, ডার্বির আগে বাড়তি অক্সিজেন পেল মোহনবাগান (**Translation:** Aghan Mahamedan defeats Bengaluru, Mohun Bagan gets extra oxygen before the derby)

Claim Label: ***Checkworthy***
Sentiment Label: ***Positive***

Ex: 2

এবার লক্ষ্মীপুজোর তোড়জোড়, পঞ্জিকা মতে কবে করবেন ধনদেবীর আরাধনা? (**Translation:** This time, Lakshmi Puja is in full swing. When will you worship Goddess of Wealth according to the calendar?)

Claim Label: ***Not-Checkworthy***
Sentiment Label: ***Neutral***

Ex: 3

দুই কলেজের সংঘর্ষে রণক্ষেত্র ঢাকা, ছোড়া হল কাঁদানে গ্যাস ও সাউন্ড গ্রেনেড (**Translation:** The battlefield was covered in a clash between two colleges, tear gas and sound grenades were fired)

Claim Label: ***Checkworthy***
Sentiment Label: ***Negative***

Figure 1: Example Bengali news headlines from the CheckSent-BN dataset with corresponding claim checkworthiness and sentiment labels (English translations provided)

1 Introduction

In the contemporary digital landscape, news consumption patterns have undergone a significant transformation, with consumers increasingly accessing information from diverse sources through mobile and online platforms, instantly. (Samuels and Mcgonical, 2020). As per a report by Reuters, the shift towards digital news consumption has been particularly pronounced in India, where 71% of users prefer digital news over traditional media in 2024[1]; this number increased to 76% in 2025[2].

Digital news organizations, on the other hand, frequently produce catchy and attention-grabbing headlines designed to maximize user engagement and encourage readers to click on their articles. These compelling headlines help boost user engagement rates and, consequently, revenue for the news organizations. However, this resulting information ecosystem, where revenue is the top-most priority, presents a significant challenge regarding the reliability and factual accuracy of news content.

Claim checkworthiness detection, the stepping stone towards fact-checking a given claim, gained significant progress in resource-rich languages such as English (Gencheva et al., 2017; Arslan et al., 2020; Nakov et al., 2018; Abumansour and Zubiaga, 2022; Majer and Šnajder, 2024) and Arabic (Jaradat et al., 2018; Abumansour

[1] https://bit.ly/reuters-digital-news-report-2024
[2] https://bit.ly/reuters-digital-news-report-2025

December 23, 2025 ©2025 Association for Computational Linguistics

and Zubiaga, 2022). However, the detection of claim-checkworthiness in resource-constrained languages, such as Bengali, remains largely unexplored, particularly in the context of Bengali news headlines.

Bengali, with over 230 million native speakers globally, holds the distinction of being the sixth most spoken language worldwide (Alam et al., 2021). As the second most widely spoken language in India and the national language in Bangladesh, it confronts significant challenges in natural language processing (NLP) research due to its complex morphological structures, extensive use of compound words, and scarcity of high-quality annotated datasets, which have hindered the development of robust NLP systems. Despite some recent efforts in Bengali NLP, a critical gap remains in comprehensive datasets that address multiple classification tasks simultaneously.

The present article focuses on developing a multi-task Bengali news headline dataset with a primary focus on claim checkworthy detection, along with an additional task of sentiment classification. The key contributions in this paper can be summarized as follows:

- We present CheckSent-BN, the first comprehensive multi-task Bengali news headline dataset, comprising approximately 11.5K samples that focus on claim-checkworthiness identification and sentiment classification. An example from the dataset is shown in Figure 1.

- We proposed a cost-effective and faster way of data annotation in the resource-constrained Bengali language, utilizing three distinct large language models (LLMs): GPT-4o-mini (OpenAI et al., 2024), GPT-4.1-mini, and Llama-4 (Touvron et al., 2023). Further, we applied a majority voting scheme and manual revision to ensure annotation quality and consistency.

- We established baseline performance by developing multi-task learning (MTL) and single-task learning (STL) frameworks utilizing pretrained multilingual transformer models, providing benchmarks for future research and exploration.

2 Related Work

This section provides an overview of recent research on claim checkworthiness detection and sentiment classification, with a specific focus on Bengali and low-resource languages.

2.1 Claim Detection

Recent advancements in artificial intelligence and NLP have enabled researchers to move from basic claim detection methods (Rosenthal and McKeown, 2012; Chakrabarty et al., 2019; Pathak et al., 2020; Gupta et al., 2021; Wührl and Klinger, 2021; Sundriyal et al., 2021; Gangi Reddy et al., 2022) to identifying the check-worthiness of claims (Jaradat et al., 2018; Wright and Augenstein, 2020). This progression has led to more advanced techniques, such as claim span identification (Sundriyal et al., 2022; Mittal et al., 2023), where specific phrases in a text that constitute a claim are pinpointed.

While significant strides have been made in high-resource languages like English, research on claim detection and claim check-worthiness identification is still limited in resource-constrained languages like Bengali. Early efforts by Dhar and Das (2021) introduced an expectation-maximization (EM) approach combined with principal component analysis (PCA) to identify claims in low-resource Indian languages, including Bengali. Their study revealed that dimensionality reduction techniques could improve classifier performance in resource-constrained contexts. Additionally, Rahman et al. (2025) created a claim detection dataset in Bengali, comprising 5,000 samples. They evaluated this dataset by incorporating traditional machine learning models with deep word embeddings, discovering that ensemble methods outperformed individual models, especially when using domain-specific Bangla embeddings.

On a broader scale, Dutta et al. (2023) released a multilingual dataset that includes English, Hindi, and Bengali, featuring factual claims extracted from Indian Twitter. Supporting efforts, such as those by Mittal et al. (2023), introduced multilingual claim span datasets based on Bengali social media texts, emphasizing span-level rather than sentence-level check-worthiness. Poddar et al. (2024) organized a shared task at the ICPR-2024 conference on multilingual claim span identification in Hindi and English tweets. Supporting this effort, Roy et al. (2025) extended the ICPR-2024 shared task dataset with more Hindi, English, Bengali, and CodeMix tweets annotated with claim spans. Recently, the CLEF-2025 CheckThat! Lab further integrated Bengali into global claim verification frameworks, which include subjectivity detection and claim normalization (Alam et al., 2025).

2.2 Sentiment Classification

In recent years, researchers have focused on more efficient transformer-based approaches. For example, Islam et al. (2020) proposed a multilingual-BERT-based sentiment classification method. The authors combined multilingual BERT embeddings with LSTM, GRU, and CNN networks, demonstrating improved performance compared to traditional embeddings such as FastText and Word2Vec. Additionally, Bhowmick and Jana (2021) developed a transformer-based sentiment analysis technique by fine-tuning two transformer models: BERT and XLM-RoBERTa. A more recent study by Pal et al. (2025) introduced a multi-task learning framework for sentiment analysis and emotion recognition in Bengali text, utilizing three transformer models: mBERT, MuRIL, and IndicBERT.

Moreover, several resources for Bengali sentiment analysis have been developed by various researchers. Islam et al. (2021) proposed 'SentNoB', a comprehensive Bengali sentiment analysis dataset consisting of approximately 15.7K noisy Bengali texts, which were curated from prominent Bangladeshi news article comments and YouTube comments. Ahmed Masum et al. (2021) created 'BAN-ABSA', an aspect-based Bengali sentiment analysis dataset with nearly 9,000 samples. In addition to dataset development, the authors established baseline frameworks and evaluated them on their dataset using Bi-LSTM and CNN techniques. Islam and Alam (2024) developed a novel dataset called 'BanglaDSA', which comprises approximately 200K Bengali comments. Along with this dataset, the authors proposed a new approach that combines Skipgram with Bangla-BERT, outperforming all existing machine learning and deep learning methods. Rashid et al. (2024) introduced another Bengali sentiment analysis dataset with a sample size of 78K by collecting reviews from two popular e-commerce websites in Bangladesh. Furthermore, Islam and Masudul Alam (2023) conducted a study focused on Bengali reviews, developing a dataset of around 44K curated reviews from restaurant Facebook pages and groups. Kabir et al. (2023) created a Bengali book review dataset, which includes approximately 158K entries.

Furthermore, Hasan et al. (2023) organized a shared task on sentiment analysis in the Bengali language at the BLP workshop, where over 25 participants submitted systems ranging from traditional machine learning approaches to pretrained transformer models, as well as state-of-the-art LLM methodologies.

Although significant progress has been made in checkworthy claim detection and sentiment classification tasks, there remains a notable research gap in creating a multi-task Bengali dataset. Additionally, the use of LLMs for data annotation in resource-constrained languages, such as Bengali, has not been extensively explored. This research aims to address this gap by developing a larger Bengali claim check-worthy dataset containing ≈11,500 samples, along with sentiment labels. This new dataset surpasses the previously established Bengali claim check-worthy dataset by Rahman et al. (2025), which included only 5,000 samples.

3 Dataset Development

The development of our dataset was conducted in three distinct phases: (1) Data Collection, (2) Data Annotation utilizing multiple LLMs, and (3) Final Label Selection via Majority Voting.

3.1 Data Collection

Data were systematically collected from prominent online Bengali news portals, specifically Bartaman[3], Sangbad Pratidin[4], Ei Samay[5], and News18 Bangla[6]. News headlines were extracted from the main news page (home page) of each portal to ensure broad coverage across various domains and topics. The Python BeautifulSoup web-scraping library was used to collect these headlines daily. The data collection period lasted from August 10, 2024, to January 12, 2025. However, data collection for the Ei Samay news portal began later, on October 5, 2024. In contrast, the collection of news from the News18 Bangla portal was discontinued after a few days due to incomplete and low-quality headlines. All collected data were stored in an Excel spreadsheet for subsequent analysis.

Following the completion of data collection, duplicate entries were removed. Subsequently, all news headlines underwent a rigorous review by three computer science interns to identify and correct any inconsistencies, such as HTML tags or unrecognized characters. Finally, a total of 11,568

[3] https://bartamanpatrika.com/

[4] https://www.sangbadpratidin.in/

[5] https://eisamay.com/

[6] https://bengali.news18.com/

unique headlines were collected from the aforementioned news portals, including 4,610 headlines from Bartaman, 5,715 from Sangbad Pratidin, 858 from Ei Samay, and 385 from News 18 Bangla.

3.2 Data Annotation

The data annotation process was carried out using three distinct LLMs. While manual annotation is considered the gold standard for creating high-quality datasets, it often encounters significant challenges, particularly in resource-limited languages such as Bengali. These challenges include: (1) a scarcity of skilled annotators, (2) the time-consuming nature of the process, and (3) the prohibitively high overall cost.

To address these issues, we adopted a cost-effective and efficient approach to data annotation by leveraging LLMs. Given the state-of-the-art performance of LLMs across various NLP tasks, we utilized three specific models: GPT-4o-mini, Llama-4, and GPT-4.1-mini. Each model was provided with a concise prompt to annotate the claim checkworthiness and sentiment labels. Each LLM was accessed via its corresponding API, and the following prompt was provided to each LLM:

```
You are an efficient language model that can do
many tasks. Now act as a professional Bengali data
annotator. You have provided a news headline in the
Bengali language. Your task is to:

1. Identify news headline claim is Checkworthy
or Not.
Checkworthy (Label = 1): Headlines that contain
factual claims needing verification
Not Checkworthy (Label = 0): Headlines that are
opinions, questions, satire, etc.

2. Identify sentiment in news headline as
'POSITIVE', 'NEGATIVE', or 'NEUTRAL'.
POSITIVE: Expresses an optimistic, successful,
complimentary, or positive outlook.
NEGATIVE: Expresses loss, criticism, failure, or a
negative outlook.
NEUTRAL: Neutral or simply informational, no emotional
coloring.

Now annotate the above-mentioned annotations in
the following news headline.
NEWS HEADLINE: {txt}

Note that only provide the exact annotation
tags in list format (Eg, [Claim Checkworthy Label,
'Sentiment Label']), no extra explanation is needed.
For clear understanding, I am providing you with some
examples.

ANNOTATION EXAMPLES:
========================
** 12 unique annotation examples were provided (See
Appendix A)**
```

This approach ensured a scalable, efficient method for annotating news headlines with claim checkworthiness and sentiment labels, which were subsequently aggregated via majority voting to select the final annotated label.

Although strict instructions were given to the LLMs to provide only claim and sentiment labels, in some instances (approximately 150 headlines), Llama-4 provided explanatory results alongside the labels. These samples were manually identified and formatted adequately by the authors.

3.3 Final Annotated Label Selection from Three LLMs' Annotated Labels

Following the annotation by three distinct LLMs, the final label for each data point was determined through majority voting. For both claim checkworthiness and sentiment, the label with the most frequent outcome was selected as the final label.

The inter-annotator agreement among the three LLM annotators was evaluated using both Fleiss' Kappa (Fleiss, 1971) and Gwet's AC1 (Gwet, 2006) metrics. For assessing the claim checkworthiness, the Fleiss' Kappa score was 0.45, while Gwet's AC1 score was 0.83. In terms of sentiment annotation, the Fleiss' Kappa score was 0.63, and Gwet's AC1 score was 0.67. These scores generally indicate a good level of agreement among the different LLM annotators.

Instances where no majority label was found through automated voting were meticulously analyzed and annotated by the authors. Additionally, all LLM-annotated data, following majority voting, underwent a thorough review by three undergraduate computer science interns. Headlines identified by the interns as inconsistent were further reviewed and resolved by the authors wherever applicable. A flow diagram of the overall data annotation process is provided in Figure 2. The distribution of claim checkworthy and sentiment labels for training and testing splits is provided in Table 1.

	Label	#Train	#Test
Claim	Check-worthy	8143	2030
	Not Check-worthy	1111	284
Sentiment	Negative	4573	1123
	Neutral	3187	825
	Positive	1494	366

Table 1: Distribution of claim checkworthiness and sentiment labels in train and test splits of CheckSent-BN.

4 Methodology

This section presents a comprehensive methodology for classifying claim-checkworthiness and sentiment for the CheckSent-BN dataset. Given

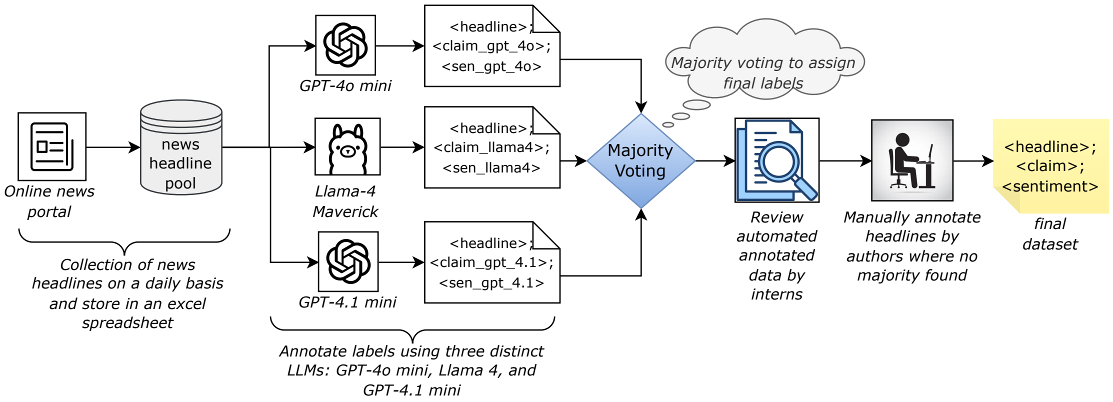

Online news portal
news headline pool
Collection of news headlines on a daily basis and store in an excel spreadsheet
GPT-4o mini
Llama-4 Maverick
GPT-4.1 mini
<headline>; <claim_gpt_4o>; <sen_gpt_4o>
<headline>; <claim_llama4>; <sen_llama4>
<headline>; <claim_gpt_4.1>; <sen_gpt_4.1>
Annotate labels using three distinct LLMs: GPT-4o mini, Llama 4, and GPT-4.1 mini
Majority Voting
Majority voting to assign final labels
Review automated annotated data by interns
Manually annotate headlines by authors where no majority found
<headline>; <claim>; <sentiment>
final dataset

Figure 2: Overview of the annotation pipeline for CheckSent-BN, including data collection, annotation with three LLMs, majority voting, and manual verification.

a news headline S, our main objective is to develop an MTL framework that can classify claims as checkworthy or not and identify the sentiment as positive, negative, or neutral in S using a single neural network. We experimented with several transformer-based pre-trained models, ranging from the lightweight multilingual DistilBERT (mDistilBERT) to Indian language (including Bengali) focused models such as IndicBERT, BanglaBERT, etc. The overall system framework is demonstrated in Figure 3.

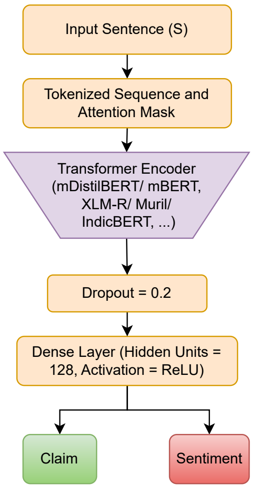

Input Sentence (S)
Tokenized Sequence and Attention Mask
Transformer Encoder (mDistilBERT/ mBERT, XLM-R/ Muril/ IndicBERT, ...)
Dropout = 0.2
Dense Layer (Hidden Units = 128, Activation = ReLU)
Claim
Sentiment

Figure 3: Flow diagram of the MTL framework. Pre-trained multilingual and Bengali-focused transformers are fine-tuned jointly for claim checkworthiness and sentiment classification.

Tokenization: Before proceeding to framework development and training, all the news headlines were tokenized into a sequence of tokens. The tokenization was performed using the corresponding pre-trained model's tokenizer, for example, for mBERT, the BERT tokenizers were used, and so on. The tokenizers were imported using the HuggingFace API. All the news headlines were tokenized to a fixed sequence length of 50 tokens, and the Input IDs (numeric representation of tokens) and attention masks were stored for further processing.

Framework Description: A diverse range of pre-trained multilingual transformer models was chosen to train and fine-tune the MTL framework, including lightweight mDistilBERT (Sanh et al., 2019), powerful mBERT (Devlin et al., 2018), XLM-RoBERTa (Conneau et al., 2019), mDeBERTa (He et al., 2021), and language-focused models such as BanglaBERT (Bhattacharjee et al., 2022) for Bengali, and MuRIL (Khanuja et al., 2021), IndicBERT (Kakwani et al., 2020), and IndicBERTv2 (Doddapaneni et al., 2023) for Indian languages. While mBERT and XLM-RoBERTa are trained on more than 100 languages, including Bengali, and excel in the majority of benchmark datasets, MuRIL, IndicBERT, and IndicBERTv2 are trained explicitly on Indian languages, including Bengali, allowing them to understand indian contexts accurately than other multilingual transformer models.

On the other hand, the BanglaBERT was explicitly trained on the Bengali language only, which allows it to understand the Bengali language more effectively than other models.

In the transformer models, the input IDs and attention masks, which were obtained from the corresponding transformer model's tokenizer function, were provided as inputs to the models. The pooler output from the transformer models, which is a learned linear transformation followed by a $tanh$ activation function applied to the last hidden state representation of the special starting token in the transformer models, was further passed through a dropout layer with a dropout rate of 0.2.

Next, the output of the dropout layer was passed through a dense layer of 128 hidden units with the ReLU activation function.

$$\mathbf{Z_{dense}} = \mathrm{ReLU}(\mathbf{Z_{dropout}})$$

Here, $\mathbf{Z_{dropout}}$ represents the output of the dropout layer, and $\mathbf{Z_{dense}}$ represents the output of the dense layer.

Classification: For multi-task classification, the output of the dense layer ($\mathbf{Z_{dense}}$) was passed through two task-specific output layers as depicted in Figure 3. The first layer was used for claim classification, employing two hidden units, and the second layer was for sentiment classification, which used three hidden units. Both the output layers used softmax as their activation function.

$$\mathcal{P}_* = softmax(\mathbf{Z_{dense}})$$
$$\hat{\mathcal{Y}}_* = \underset{j}{argmax}(P_*)$$

Here, $\mathcal{P}_*$ denotes the probability value for each class in a classification task, $\hat{\mathcal{Y}}_*$ indicates the predicted class value, and j represents the number of classes involved in the classification task.

Training: To train the framework, the training data was first divided into a 9:1 ratio, where 90% of the data was used for training and the remaining 10% was used as a validation set. The `SparseCategoricalCrossEntropy` loss function was used during training, and the loss was monitored for the validation set.

$$\mathcal{L}_{total} = \mathcal{L}_{claim} + \mathcal{L}_{sentiment}$$

Here, $\mathcal{L}_{claim}$ and $\mathcal{L}_{sentiment}$ represent the loss for claim checkworthiness and sentiment classification tasks, and $\mathcal{L}_{total}$ is the overall loss function. The primary objective during training was to minimize the total loss ($\mathcal{L}_{total}$) for the validation data.

The AdamW optimizer (Loshchilov and Hutter, 2019) was selected for optimization, utilizing a weight decay of 0.01 and an epsilon value of 1e-8. The learning rate was set to 2e-5, and the frameworks were trained for a maximum of ten epochs with a batch size of 16.

5 Experiment and Result

This section provides a brief overview of the experimental setup and the outcomes obtained from the experiments.

5.1 Experimental Setup

All experiments were conducted using TensorFlow and Keras, and the models were trained on an NVIDIA RTX 5000 GPU. The experiments were employed in two setups: an MTL configuration, where the tasks of claim checkworthiness identification and sentiment classification shared the same neural network with two classification heads, and a single-task learning (STL) configuration, where separate neural networks were developed for each task, with each network featuring a single classification head. A diagrammatic representation of the STL framework is provided in Appendix C.

To ensure a fair comparison, all MTL and STL frameworks were trained with the same hyperparameters as mentioned in Section 4. For evaluation, the precision, recall, and macro F1-score were calculated on the test dataset.

5.2 Result

Claim Checkworthiness Identification: The results of the claim checkworthiness identification are summarized in Table 2. It is observed from this table that the IndicBERTv2 model excels all other models in both the STL and MTL frameworks, achieving F1-scores of 82.91 and 83.86, respectively. Additionally, the majority of transformer models (mDistilBERT, mBERT, mDeBERTa, BanglaBERT, IndicBERT, and IndicBERTv2) demonstrate improved claim identification results with the MTL framework compared to their STL counterparts.

Notably, the IndicBERTv2 model exhibits a performance enhancement of 1.13% in the MTL framework compared to its STL framework. The other transformer model-based multi-task learning frameworks, including mDistilBERT, mBERT, mDeBERTa, BanglaBERT, and IndicBERT, show performance improvements of 2.07%, 5.15%, 1.05%, 2.48%, and 0.09%, respectively, compared to their corresponding single-task learning frameworks. However, a slight decline in performance is observed for the XLM-RoBERTa and MuRIL models in the MTL framework, with F1-score drops of 1.69% and 2.65%, respectively.

Sentiment Classification: The results for sentiment classification are presented in Table 3. Similarly, for claim checkworthiness identification in the context of sentiment classification, the IndicBERTv2 model outperforms other transformer models. Moreover, the IndicBERTv2-based MTL

	STL			MTL		
	Prec.	**Rec.**	**F1.**	**Prec.**	**Rec.**	**F1.**
mDistilBERT	78.64	69.37	72.73	73.69	74.90	74.27
mBERT	77.61	68.12	71.45	77.30	73.74	75.33
XLM-R	79.42	77.61	78.47	82.97	73.52	77.14
mDeBERTa	84.80	78.37	81.13	83.72	80.49	81.99
BanglaBERT	87.91	75.77	80.27	**86.21**	79.40	82.31
MuRIL	83.57	76.07	79.15	85.23	72.63	77.06
IndicBERT	79.55	65.29	69.30	79.81	65.31	69.36
InidcBERTv2	86.13	80.38	82.91	85.48	**82.43**	**83.86**

Table 2: Performance of transformer models on claim checkworthiness detection under STL and multi-task MTL frameworks. (Best Precision, Recall, and F1-score are in bold font)

framework demonstrates an improvement of 1.26% in terms of F1-score compared to the IndicBERTv2-based STL framework.

Regarding other transformer models, XLM-RoBERTa, mDeBERTa, BanglaBERT, MuRIL, and IndicBERT yielded better results within the MTL framework, with F1-score enhancements of 0.93%, 0.45%, 1.53%, 1.3%, and 0.47%, respectively, compared to their corresponding STL counterparts. In contrast, mDistilBERT and mBERT did not perform as effectively for sentiment classification in the multi-task learning scenario, resulting in similar or lower performance compared to the STL frameworks, with performance drops of 0.87% and 0.38%, respectively.

	STL			MTL		
	Prec.	**Rec.**	**F1.**	**Prec.**	**Rec.**	**F1.**
mDistilBERT	65.33	66.24	65.46	64.75	65.36	64.89
mBERT	71.43	66.15	67.37	69.42	66.07	67.11
XLM-R	74.07	74.73	74.35	74.84	75.30	75.05
mDeBERTa	72.02	78.37	78.64	79.91	78.27	79.00
BanglaBERT	82.63	77.62	78.98	80.00	80.11	80.05
MuRIL	75.73	77.40	75.79	76.23	77.51	76.79
IndicBERT	64.03	61.09	61.78	65.61	60.85	62.07
IndicBERTv2	80.39	**81.09**	80.82	**83.70**	80.02	**81.52**

Table 3: Performance of transformer models on sentiment classification under STL and MTL frameworks. (Best Precision, Recall, and F1-score are in bold font)

5.3 Observations

Upon performing all the experiments and analysing the results, a few observations are made:

First, it is observed that for both checkworthy claim identification and sentiment classification, the IndicBERTv2 demonstrates impressive results, irrespective of the STL and MTL frameworks. Additionally, the BanglaBERT and mDeBERTa models indicate a strong performance compared to other transformer models. This improvement suggests a better contextual understanding of the Bengali language compared to other transformer models.

Second, the joint learning of claim-checkworthiness detection and sentiment classification leads to more effective identification of claim-checkworthy sentences than using STL classifiers in most cases. This observation shows that the sentiment classification task effectively assists in identifying claim checkworthiness within an MTL environment.

Third, the performance of the IndicBERT model across all tasks and frameworks is relatively low. One possible reason for this is that IndicBERT was trained on the ALBERT model using 12 Indian languages, resulting in a smaller and more lightweight model compared to other transformer models, such as mBERT, MuRIL, and XLM-RoBERTa. As a result, it may struggle to identify nuanced contexts in text, leading to lower performance compared to other transformer-based frameworks.

6 Error Analysis

Due to a diverse range of experiments with different transformer-based models in both MTL and STL frameworks, the error analysis for each model is a challenging task. Therefore, to simplify the error analysis, we conducted the error analysis between the best-performing models (i.e., IndicBERTv2) for both MTL and STL frameworks. To conduct the error analysis, we calculated the confusion matrices for each task in each framework.

Claim Checkworthiness Detection: The confusion matrices for claim checkworthiness detection are provided in Figure 4. Although the IndicBERTv2-based MTL framework demonstrates strong overall performance, it slightly lacks in identifying claim-checkworthy sentences. Out of 2030 claim checkworthy instances, the MTL framework identifies 96.9% sentences as claim checkworthy, whereas the STL identifies claim checkworthy texts with 97.4% accuracy.

On the other hand, the STL framework correctly identifies 63.4% of the 284 non-claim checkworthy instances, while the MTL framework achieves 68%, demonstrating better performance.

Sentiment Classification: The confusion matrices for sentiment classification for IndicBERTv2-based STL and MTL frameworks are provided in Figure 5. The confusion matrices indicate that

ID	News Headline	True Label		Predicted STL		Predicted MTL	
		Claim	Sentiment	Claim	Sentiment	Claim	Sentiment
S_1	বছর শেষে ছুটির আমেজ, ভিড়ে ঠাসা দিঘা থেকে দার্জিলিং, দেখুন ছবি (**Translation:** End of year holiday atmosphere, crowded Digha to Darjeeling, see photos)	n-claim	neutral	claim	positive	n-claim	positive
S_2	‘সিনেমাপাড়ার একটাই স্বর, জাস্টিস ফর RG Kar’, পথে নামল টলিউড (**Translation:** 'Cinemapara has one voice, Justice for RG Kar', Tollywood takes to the road)	claim	neutral	no-claim	positive	claim	neutral
S_3	খসছে পদ্মের পাঁপড়ি! উপনির্বাচনে ৬-এ শূন্য পেয়ে কত দাঁড়াল বিজেপির বিধায়ক সংখ্যা? (**Translation:** The lotus petals are falling! How many MLAs did BJP have after getting zero in 6 by-elections?)	claim	negative	claim	neutral	claim	negative
S_4	২ লক্ষ কোটি টাকার কুম্ভ ইকনমি! হিন্দুত্বকে সামনে রেখে ঢালাও ব্যবসা যোগীরাজ্যে, আশায় বুক বাঁধছে বণিকসভা (**Translation:** Kumbh Economy of 2 lakh crore rupees! Keeping Hindutva in the forefront, business will pour into Yogi Rajya, the Chamber of Commerce is full of hope)	claim	positive	claim	positive	claim	negative

Table 4: Example misclassifications from IndicBERTv2 models in STL and MTL settings. Each row displays the news headline, gold labels (true labels), and predicted labels (with an English translation provided).

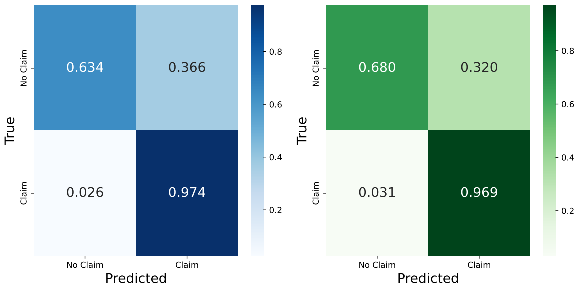


Figure 4: Confusion matrices for claim checkworthiness detection using IndicBERTv2 in STL and MTL setups.

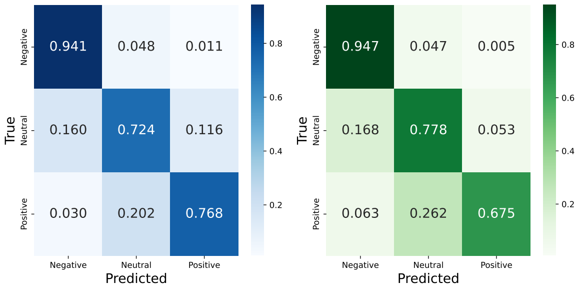


Figure 5: Confusion matrices for sentiment classification using IndicBERTv2 in STL and MTL setups.

the IndicBERTv2-based STL framework achieves 94.1% and 72.8% accuracy in identifying negative and neutral sentiments, respectively. In contrast, the IndicBERTv2-based MTL framework achieves accuracies of 94.7% and 77.8%, respectively, resulting in a decrease in the number of misclassified cases for these categories.

Conversely, the STL framework outperforms the MTL framework in identifying positive sentiments, achieving an accuracy of 76.8%, compared to the MTL framework's 67.5%. This suggests that STL is more effective at recognizing positive sentiments, although MTL exhibits better performance in the negative and neutral classes.

Along with the confusion matrix, a few examples of error cases are provided in Table 4. For instance, in example S_1, whereas the original claim was non-checkworthy, the IndicBERTv2-based STL framework incorrectly classified it as claim-checkworthy. Additionally, both STL and MTL frameworks mistakenly classify the sentiment as ‘positive’ where the original sentiment was ‘neutral’. While the news headline in S_1 carries a slight positive emotional tone, it primarily describes a factual situation: “*year-end holiday atmosphere*”. The phrase “*crowded Digha to Darjeeling*” is a neutral description of a high-traffic situation, not necessarily negative or positive. Therefore, the ‘neutral’ sentiment is justified for the headline. However, neither the STL nor the MTL framework captures these contextual nuances in the text and incorrectly categorizes it as ‘positive’.

Considering another example, S_3, the original sentiment in the news headline is negative, indicating a poor performance of the BJP (a political party in India) in the bi-election results. In this case, the STL framework incorrectly predicts the text as “negative,” whereas the MTL framework successfully identifies its sentiment label. This suggests that the joint learning of claim, sentiment, and news content enables the MTL framework to determine the sentiment label accurately.

7 Conclusion

This paper proposes ‘CheckSent-BN’, a dataset comprising 11,568 instances annotated with labels for two distinct tasks: claim checkworthiness identification and sentiment classification. We developed two baseline frameworks: the STL framework, with one classification head for each task, and the MTL framework, which has two classification heads. We experimented with eight different multilingual transformer models, and the experimental results show that the IndicBERTv2, BanglaBERT, and mDeBERTa model-based frameworks demonstrate a strong performance over all classification tasks. Notably, the IndicBERTv2-based MTL framework achieved the best performance across all classification tasks.

Future directions include expanding the dataset's sample size, comparing LLM annotations with human annotations, and adding labels such as 'clickbait' or 'sarcasm' to enhance the dataset's scope.

Limitations

Our proposed work has several potential limitations. First, the annotation of claim checkworthy and sentiment labels was conducted using three LLMs. While we performed a superficial manual verification with three computer science interns, relying on LLMs for labeling may compromise the overall quality of the dataset. In future work, we aim to hire professional Bengali data annotators to ensure more accurate verification of these labels.

Second, we utilized the mini variants of the GPT models, specifically GPT-4o-mini and GPT-4.1-mini, for cost-effectiveness. Although these models can adequately annotate claim checkworthy and sentiment labels, the "mini" variants do not fully leverage the capabilities of the full GPT models.

Third, there is a significant imbalance in the claim checkworthy labels: 9,254 are labeled checkworthy, while only 2,314 are labeled non-checkworthy. This imbalance can lead to bias in our MTL and STL frameworks due to the predominance of annotated data. We plan to address this issue in future work by developing a more balanced dataset.

Fourth, the news headlines used in our study were curated from prominent news websites focused on the state of West Bengal, India. However, there are other Bengali-speaking regions in India, such as Tripura and parts of Assam, that have their own regional newspapers in Bengali, which are not included in our current work. Additionally, news headlines from Bangladeshi news portals were also excluded. In future work, we intend to incorporate Bengali news headlines from these other areas, including Tripura and Bangladesh, to broaden the dataset's scope.

Acknowledgment

This work was supported by the Defence Research and Development Organisation (DRDO), New Delhi, under the project "Claim Detection and Verification using Deep NLP: an Indian perspective".

References

Amani S. Abumansour and Arkaitz Zubiaga. 2022. Check-worthy claim detection across topics for automated fact-checking. *Preprint*, arXiv:2212.08514.

Mahfuz Ahmed Masum, Sheikh Junayed Ahmed, Ayesha Tasnim, and Md. Saiful Islam. 2021. Banabsa: An aspect-based sentiment analysis dataset for bengali and its baseline evaluation. In *Proceedings of International Joint Conference on Advances in Computational Intelligence*, pages 385–395, Singapore. Springer Singapore.

Firoj Alam, Arid Hasan, Tanvirul Alam, Akib Khan, Janntatul Tajrin, Naira Khan, and Shammur Absar Chowdhury. 2021. A review of bangla natural language processing tasks and the utility of transformer models. *Preprint*, arXiv:2107.03844.

Firoj Alam, Julia Maria Struß, Tanmoy Chakraborty, Stefan Dietze, Salim Hafid, Katerina Korre, Arianna Muti, Preslav Nakov, Federico Ruggeri, Sebastian Schellhammer, Vinay Setty, Megha Sundriyal, Konstantin Todorov, and Venktesh V. 2025. The clef-2025 checkthat! lab: Subjectivity, fact-checking, claim normalization, and retrieval. *Preprint*, arXiv:2503.14828.

Fatma Arslan, Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2020. A benchmark dataset of check-worthy factual claims. *Preprint*, arXiv:2004.14425.

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.

Anirban Bhowmick and Abhik Jana. 2021. Sentiment analysis for Bengali using transformer based models. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 481–486, National Institute of Technology Silchar, Silchar, India. NLP Association of India (NLPAI).

Tuhin Chakrabarty, Christopher Hidey, and Kathy McKeown. 2019. IMHO fine-tuning improves claim detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages

558–563, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Rudra Dhar and Dipankar Das. 2021. Leveraging expectation maximization for identifying claims in low resource Indian languages. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 307–312, National Institute of Technology Silchar, Silchar, India. NLP Association of India (NLPAI).

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.

Subhabrata Dutta, Rudra Dhar, Prantik Guha, Arpan Murmu, and Dipankar Das. 2023. A multilingual dataset for identification of factual claims in indian twitter. In *Proceedings of the 14th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '22, page 88–92, New York, NY, USA. Association for Computing Machinery.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Revanth Gangi Reddy, Sai Chetan Chinthakindi, Yi R. Fung, Kevin Small, and Heng Ji. 2022. A zero-shot claim detection framework using question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6927–6933, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. A context-aware approach for detecting worth-checking claims in political debates. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 267–276, Varna, Bulgaria. INCOMA Ltd.

Shreya Gupta, Parantak Singh, Megha Sundriyal, Md. Shad Akhtar, and Tanmoy Chakraborty. 2021. LESA: Linguistic encapsulation and semantic amalgamation based generalised claim detection from online content. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3178–3188, Online. Association for Computational Linguistics.

Kilem Li Gwet. 2006. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.

Md. Arid Hasan, Firoj Alam, Anika Anjum, Shudipta Das, and Afiyat Anjum. 2023. BLP-2023 task 2: Sentiment analysis. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 354–364, Singapore. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Khondoker Ittehadul Islam, Md Saiful Islam, and Md Ruhul Amin. 2020. Sentiment analysis in bengali via transfer learning using multi-lingual bert. In *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, pages 1–5.

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.

Md. Shymon Islam and Kazi Masudul Alam. 2024. Sentiment analysis of bangla language using a new comprehensive dataset bangdsa and the novel feature metric skipbangla-bert. *Natural Language Processing Journal*, 7:100069.

Md. Shymon Islam and Kazi Masudul Alam. 2023. Sentiment analysis on bangla food reviews using machine learning and explainable nlp. In *2023 26th International Conference on Computer and Information Technology (ICCIT)*, pages 1–6.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 26–30, New Orleans, Louisiana. Association for Computational Linguistics.

Mohsinul Kabir, Obayed Bin Mahfuz, Syed Rifat Raiyan, Hasan Mahmud, and Md Kamrul Hasan. 2023. Banglabook: A large-scale bangla dataset for sentiment analysis from book reviews. *Preprint*, arXiv:2305.06595.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite:

Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*.

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. *Preprint*, arXiv:2103.10730.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Laura Majer and Jan Šnajder. 2024. Claim check-worthiness detection: How well do LLMs grasp annotation guidelines? In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 245–263, Miami, Florida, USA. Association for Computational Linguistics.

Shubham Mittal, Megha Sundriyal, and Preslav Nakov. 2023. Lost in translation, found in spans: Identifying claims in multilingual social media. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3887–3902, Singapore. Association for Computational Linguistics.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the clef-2018 checkthat! lab on automatic identification and verification of political claims. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 372–387, Cham. Springer International Publishing.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 9 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Pritam Pal, Dipankar Das, and Anup Kumar Kolya. 2025. Bilingual sentiment and emotion analysis: A multi-task learning framework for bengali and english. In *Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '24, page 61–66, New York, NY, USA. Association for Computing Machinery.

Archita Pathak, Mohammad Abuzar Shaikh, and Rohini Srihari. 2020. Self-supervised claim identification for automated fact checking. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 213–227, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLPAI).

Soham Poddar, Biswajit Paul, Moumita Basu, and Saptarshi Ghosh. 2024. ICPR 2024 Competition on Multilingual Claim-Span Identification (ICPR-CSI 2024).

Md. Rashadur Rahman, Rezaul Karim, Mohammad Shamsul Arefin, Pranab Kumar Dhar, Gahangir Hossain, and Tetsuya Shimamura. 2025. Facilitating automated fact-checking: a machine learning based weighted ensemble technique for claim detection. *Discover Applied Sciences*, 7(1):73.

Mohammad Rifat Ahmmad Rashid, Kazi Ferdous Hasan, Rakibul Hasan, Aritra Das, Mithila Sultana, and Mahamudul Hasan. 2024. A comprehensive dataset for sentiment and emotion classification from bangladesh e-commerce reviews. *Data in Brief*, 53:110052.

Sara Rosenthal and Kathleen McKeown. 2012. Detecting opinionated claims in online discussions. In *2012 IEEE Sixth International Conference on Semantic Computing*, pages 30–37.

Rudra Roy, Pritam Pal, Dipankar Das, Saptarshi Ghosh, and Biswajit Paul. 2025. Enhancing textual understanding: Automated claim span identification in english, hindi, bengali, and codemix. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing - Natural Language Processing in the Generative AI era*, pages 1030–1035, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Antony Samuels and John Mcgonical. 2020. News sentiment analysis. *Preprint*, arXiv:2007.02238.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Megha Sundriyal, Atharva Kulkarni, Vaibhav Pulastya, Md. Shad Akhtar, and Tanmoy Chakraborty. 2022. Empowering the fact-checkers! automatic identification of claim spans on Twitter. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7701–7715, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

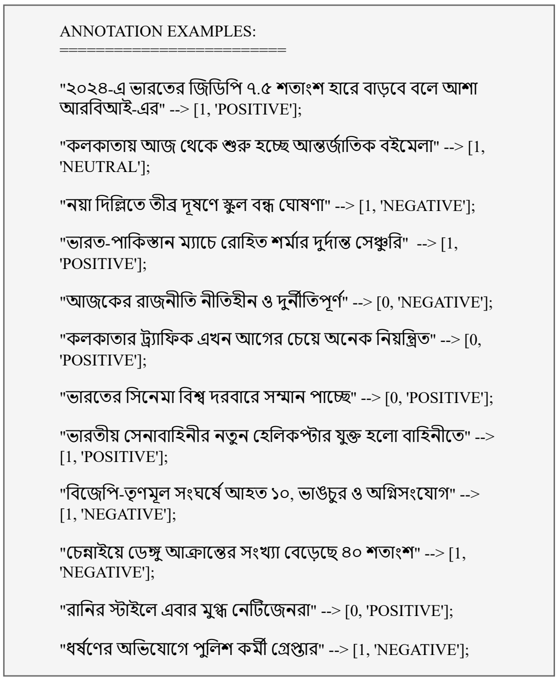
Megha Sundriyal, Parantak Singh, Md. Shad Akhtar, Shubhashis Sengupta, and Tanmoy Chakraborty. 2021. Desyr: Definition and syntactic representation based claim detection on the web. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Dustin Wright and Isabelle Augenstein. 2020. Claim check-worthiness detection as positive unlabelled learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 476–488, Online. Association for Computational Linguistics.

Amelie Wührl and Roman Klinger. 2021. Claim detection in biomedical Twitter posts. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 131–142, Online. Association for Computational Linguistics.

A Examples Used During Data Annotation by LLM



ANNOTATION EXAMPLES:
=========================

"২০২৪-এ ভারতের জিডিপি ৭.৫ শতাংশ হারে বাড়বে বলে আশা আরবিআই-এর" --> [1, 'POSITIVE'];

"কলকাতায় আজ থেকে শুরু হচ্ছে আন্তর্জাতিক বইমেলা" --> [1, 'NEUTRAL'];

"নয়া দিল্লিতে তীব্র দূষণে স্কুল বন্ধ ঘোষণা" --> [1, 'NEGATIVE'];

"ভারত-পাকিস্তান ম্যাচে রোহিত শর্মার দুর্দান্ত সেঞ্চুরি" --> [1, 'POSITIVE'];

"আজকের রাজনীতি নীতিহীন ও দুর্নীতিপূর্ণ" --> [0, 'NEGATIVE'];

"কলকাতার ট্র্যাফিক এখন আগের চেয়ে অনেক নিয়ন্ত্রিত" --> [0, 'POSITIVE'];

"ভারতের সিনেমা বিশ্ব দরবারে সম্মান পাচ্ছে" --> [0, 'POSITIVE'];

"ভারতীয় সেনাবাহিনীর নতুন হেলিকপ্টার যুক্ত হলো বাহিনীতে" --> [1, 'POSITIVE'];

"বিজেপি-তৃণমূল সংঘর্ষে আহত ১০, ভাঙচুর ও অগ্নিসংযোগ" --> [1, 'NEGATIVE'];

"চেন্নাইয়ে ডেঙ্গু আক্রান্তের সংখ্যা বেড়েছে ৪০ শতাংশ" --> [1, 'NEGATIVE'];

"রানির স্টাইলে এবার মুগ্ধ নেটিজেনরা" --> [0, 'POSITIVE'];

"ধর্ষণের অভিযোগে পুলিশ কর্মী গ্রেপ্তার" --> [1, 'NEGATIVE'];

Figure A.1: Illustrative prompts used for claim check-worthiness detection and sentiment annotation in Bengali news headlines. The set comprises 12 examples spanning diverse domains, including politics, sports, entertainment, the economy, and social issues. These examples were provided to LLMs during the annotation phase to guide consistent labeling across both tasks.

B Statistical Distribution of Data

Label	Max	Min	Mean	Median	Mode	St. Dev.
Non-Checkworthy	37	6	10.192	10	10	2.802
Checkworthy	38	6	10.188	10	10	2.477
Negative	35	6	10.264	10	10	2.338
Neutral	37	6	10.032	10	10	2.558
Positive	38	6	10.298	10	10	2.921

Table B.1: Statistical distribution of the number of words across claim checkworthiness detection (Non-checkworthy, Checkworthy) and sentiment classification (Negative, Neutral, Positive) labels.

C STL Framework (Flow Diagram)

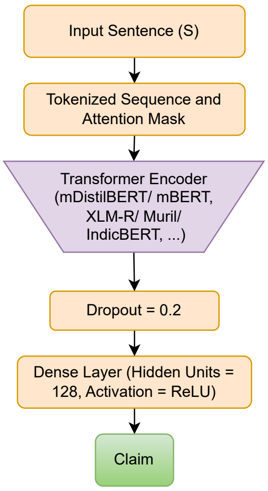

Input Sentence (S)
Tokenized Sequence and Attention Mask
Transformer Encoder (mDistilBERT/ mBERT, XLM-R/ Muril/ IndicBERT, ...)
Dropout = 0.2
Dense Layer (Hidden Units = 128, Activation = ReLU)
Claim

Figure C.1: Diagrammatic representation of the STL framework for identifying claim checkworthiness. The framework is identical to the MTL framework. However, instead of using two classification heads in the MTL framework, the STL framework employs a single classification head for each task.

Advancing Subjectivity Detection in Bengali News Articles Using Transformer Models with POS-Aware Features

Md Minhazul Kabir, Kawsar Ahmed
Mohammad Ashfak Habib and **Mohammed Moshiul Hoque**
Department of Computer Science and Engineering
Chittagong University of Engineering & Technology, Chattogram-4349, Bangladesh
{u1904040, u1804017}@student.cuet.ac.bd
{ashfak, moshiul_240}@cuet.ac.bd

Abstract

Distinguishing fact from opinion in text is a nuanced but essential task, particularly in news articles where subjectivity can influence interpretation and reception. Identifying whether content is subjective or objective is critical for sentiment analysis, media bias detection, and content moderation. However, progress in this area has been limited for low-resource languages such as Bengali due to a lack of benchmark datasets and tools. To address these constraints, this work presents **BeNSD** (Bengali News Subjectivity Detection), a novel dataset of 8,655 Bengali news article texts, along with an enhanced transformer-based architecture (*POS-Aware-MuRIL*) that integrates parts-of-speech (POS) features with MuRIL embeddings at the input level to provide richer contextual representation for subjectivity detection. A range of baseline models is evaluated, and the proposed architecture achieves a macro F1-score of 93.35% in subjectivity detection for the Bengali language. The code of the work is available on GitHub[1].

1 Introduction

The expansion of digital platforms and online journalism has resulted in a substantial increase in text-based content. News articles, blogs, and social media posts now serve as primary sources of information for a global audience. These media have a significant impact on public discourse and opinion formation. The proliferation of digital information has also introduced challenges, including the dissemination of biased, subjective, and manipulative content that complicates the consumption of accurate information and shapes public sentiment. Transitions within a single sentence from factual reporting to personal opinion can significantly alter the reader's interpretation and perception. Subjectivity detection in news articles refers to the process of distinguishing between factual statements and those that express opinions, emotions, or personal judgments. Subjectivity detection helps identify opinion-based statements, supporting balanced reporting and thereby enhancing the credibility of news sources.

Effective subjectivity detection supports news credibility, mitigates bias, and promotes the distribution of impartial information (Satapathy et al., 2022). Recent advancements in subjectivity detection have primarily been observed in high-resource languages, including English and other European languages. In contrast, subjectivity detection in Bengali remains understudied due to limited linguistic resources. As the volume of Bengali content on digital platforms grows, the need for effective subjectivity detection becomes increasingly important. However, progress is hindered by a scarcity of public datasets and the complexity of Bengalis syntax, morphology, and annotation ambiguity. This work contributes in the following ways to address current constraints in subjectivity detection in Bengali:

- We introduce **BeNSD**, a new annotated dataset comprising 8,655 Bengali news sentences, each labelled as either subjective (SUBJ) or objective (OBJ)
- We propose a transformer-based model that leverages MuRIL language model with POS embeddings. By integrating syntactic features with deep contextual representations, the model enhances its ability to recognize the subtle linguistic markers that distinguish subjective from objective expressions in Bengali texts.

2 Related Work

Subjectivity detection in news articles has seen marked progress in high-resource languages such

[1] https://github.com/R1FA7/Subjectivity-Detection-in-Bengali-News-Articles

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 131–141
December 23, 2025 ©2025 Association for Computational Linguistics

as English. Paran et al. (2024) developed a model for subjectivity detection using datasets of 1,776 English and 2,675 Arabic news article sentences. Their approach achieved the highest F1 Scores of 72.6% on Arabic and 50.36% on English using Llama-3-8b, with the dataset size potentially contributing to the results. Antici et al. (2023) introduced annotation guidelines and a corpus of 1,049 annotated sentences for subjectivity detection in English news articles. Both monolingual and multilingual classification setups were examined. In the monolingual setting, m-BERT achieved macro-F1 scores of 75% for English and 74% for Italian. The multilingual use of m-BERT yielded 5% (for English) and 3% (for Italian) improvements in macro F1 scores. Pachov et al. (2023) applied a dataset of 1,019 English sentences, with a majority voting ensemble achieving the highest macro F1 score of 0.77. Frick (2023) studied subjectivity classification using LLM-augmented data with datasets of 1,292 English and 1,291 German sentences, employing BERT, GPT, and their combination. Their approach yielded an F1 value of 0.73, but performance was inconsistent when ChatGPT was used in few-shot settings. Dey et al. (2023) used a multilingual dataset in six languagesEnglish, Arabic, Dutch, German, Italian, and Turkishand applied transformer architectures, including BERT, M-BERT, and XLM-RoBERTa. The XLM-RoBERTa large model achieved the highest F1 score of 0.82 on a dataset comprising 7,828 texts. Additionally, a BERT-based multitask learning framework combining sentiment and subjectivity detection was introduced by Satapathy et al. (2022), with the addition of a Neural Tensor Network resulting in a 24% absolute improvement in accuracy, reaching 95.1%. On the IMDB dataset of 10,000 English movie reviews, A recent work Sagnika et al. (2021) proposed an opinion mining technique for subjectivity identification using an attention-based CNN-LSTM model, achieving an accuracy of 0.971.

In contrast to the progress made in high-resource languages, subjectivity detection in low-resource languages remains in a rudimentary stage. Suwaileh et al. (2025) proposed a dataset for subjectivity detection in Arabic news sentences, comprising 3,600 manually annotated sentences named ThatiAR. Their framework leveraged various transformers and LLMs, but using the 3-shot and 5-shot settings of GPT-4, they achieved the highest weighted F1 score of 0.80. Chaturvedi et al. (2017) proposed a framework using an extreme learning machine with Bayesian networks and fuzzy recurrent neural networks (FRNNs) for subjectivity detection and achieved an accuracy of 89%. A recent study (Dwivedi et al., 2024) explored subjectivity analysis in nine low-resource Indian languages using GPT-4 and BARD through in-context learning and prompt engineering. They showed that language-specific prompts significantly improve performance in multilingual settings. Around 7,000 domain-specific sentences were collected, and the data was balanced for subjective and objective classes by Dwivedi and Ghosh, 2022. They designed a lexical-rule-based Finite State Transducer (FST) for five Indian languages: Bengali, Hindi, Odia, Khorti, and Kannauji. Their system achieved an average accuracy of 84% across five languages.

Differences with existing research: Although notable progress has been made in subjectivity detection across high-resource languages, a significant gap remains in Bengali. In our exploration, no benchmark dataset is available for subjectivity detection in Bengali news articles, and no prior systems have been proposed for this task. Furthermore, previous methods have often overlooked the role of syntactic structures, such as POS, in shaping subjectivity. Instead, they have relied primarily on semantic features. To address these gaps, this work differs from existing studies in two crucial ways: (i) it introduces the first large-scale, manually annotated dataset, **BeNSD** for subjectivity detection in Bengali news articles, and (ii) it proposes a *POS-Aware* transformer-based model, combining syntactic and contextual cues, to improve subjectivity detection in Bengali.

3 Development of Dataset: BeNSD

This work develops **BeNSD**, a dataset for detecting subjectivity in news articles, as benchmark datasets are currently unavailable in Bengali. In news media, it is often hard to distinguish between subjective and objective text as the differences can be subtle. A *subjective text* expresses personal opinions, emotions, speculations, rhetorical questions, sarcasm, exaggerations, or unsupported conclusions. In contrast, the *objective text* presents information neutrally, reports events as they happen, includes third-party statements, and uses data-supported conclusions (Antici et al., 2021).

3.1 Data Collection and Preprocessing

Bengali news articles were collected through a combination of manual browsing (42 articles) and web scraping (251 articles) from prominent Bengali news portals. To ensure data diversity, sources were selected to represent a wide range of topics, including politics, health, sports, entertainment, and social issues. Initially, 8,800 raw news texts were gathered, with the majority (8,144) collected from Prothom Alo and the least amount (37) from Jugantor. The accumulated data was collected between January 21, 2025, and April 14, 2025. Figure 1 shows the source-wise distribution of collected data.

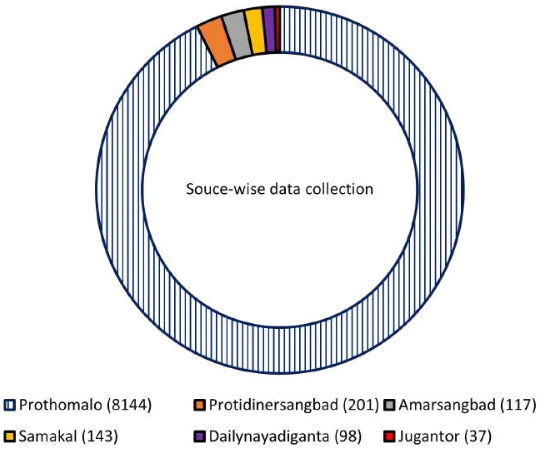


Figure 1: Source-wise accumulated data distribution. The values in the legends indicate the amount of data collected from each source.

To reduce manual annotation effort and redundancy, cleaning and filtering steps were applied: non-Bengali characters, extra punctuation, and special symbols were removed; duplicates were discarded; and articles with fewer than three meaningful words were filtered out. After preprocessing, 115 texts were removed, leaving 8,685 valid texts for manual annotation.

3.2 Data Annotation

After preprocessing, 8,685 news texts were given to annotators to label as *subjective (SUBJ)* or *objective (OBJ)*. Three undergraduate computer science students independently labelled each article, and their work was reviewed by an expert with over 20 years of experience in NLP research. Annotators followed clear class definitions to ensure consistent labelling, and all texts were annotated independently, without regard to prior context. Initially, they determined whether each sentence was subjective or objective by applying the provided guidelines. Before starting the main task, they practiced on a small set of example sentences to ensure they understood the difference between the two classes. This training helped clarify how they interpreted the definitions. Annotators also wrote brief justifications for their choices, which helped resolve disagreements during expert review. Since each text was labelled by three annotators, the final class label was decided through majority voting. After annotation, the expert removed 30 texts because their tone was ambiguous. The final dataset includes 8,655 annotated texts.

Raw Text	Processed Text	Remarks
বর্ণিত ম্যাক্স-মিন পদ্ধতি (Max-min principle), তথা সর্বাধিক দলের ন্যূনতম জরুরি সংস্কারে একমত হওয়া একটি বাস্তবসম্মত পথ তৈরি করতে পারে	বর্ণিত ম্যাক্স-মিন পদ্ধতি, তথা সর্বাধিক দলের ন্যূনতম জরুরি সংস্কারে একমত হওয়া একটি বাস্তবসম্মত পথ তৈরি করতে পারে	Removed non-Bengali characters
একটি ছেলে — যে এখানেই বড় হয়েছে... সবকিছু জিতেছে!	একটি ছেলে যে এখানেই বড় হয়েছে সবকিছু জিতেছে	Remove special symbol & punctuation
অনেকেই এসেছেন	Discarded	Text fewer than three words

Table 1: Preprocessing examples.

To evaluate annotation quality, we measured inter-annotator agreement using Cohens kappa coefficient (Cohen, 1960). The resulting kappa score was 0.92, indicating almost perfect agreement and demonstrating the reliability of the annotated dataset. The finalized dataset was then converted into a standardized format (e.g., Excel) for further processing.

3.3 Dataset Statistics

The **BeNSD** dataset is randomly split into training (80%), validation (10%), and test (10%) sets, with label stratification to maintain class balance for model development and evaluation. Table 2 provides a summary of the developed dataset. It is observed that OBJ samples have more words than SUBJ samples across all sets. On average, each text contains 13 to 14 words. The validation and test sets exhibit significantly higher lexical diversity (approximately 0.5) than the training set (approximately 0.24), indicating that they use a broader vocabulary despite being smaller.

We conducted several statistical analyses to better understand the dataset's characteristics. Figure 2 illustrates the distribution of SUBJ and OBJ text

Set	Class	Samples	W_T	W_U	W_{Avg}	W_U/W_T
Train	OBJ	4007	56,321	13,704	14.06	0.2433
	SUBJ	2917	38,629	9,568	13.24	0.2477
Subtotal		6924	94,950	18,859	-	-
Val	OBJ	501	6,895	3,539	13.76	0.5133
	SUBJ	364	4,900	2,445	13.46	0.4990
Subtotal		865	11,795	5,123	-	-
Test	OBJ	501	7,176	3,680	14.32	0.5128
	SUBJ	365	4,819	2,370	13.20	0.4918
Subtotal		866	11,995	5,173	-	-
Total		8,655	118,740	29,155	-	-

Table 2: Dataset statistics across training, validation, and test splits, where W_T, W_U, W_{Avg}, and W_U/W_T denote the total word count, unique word count, average words per sample, and lexical diversity ratio, respectively.

lengths in the dataset. The diagram shows that most OBJ sentences fall within the 7- to 15-word range, indicating a concentration of short news pieces. SUBJ sentences follow a similar pattern.

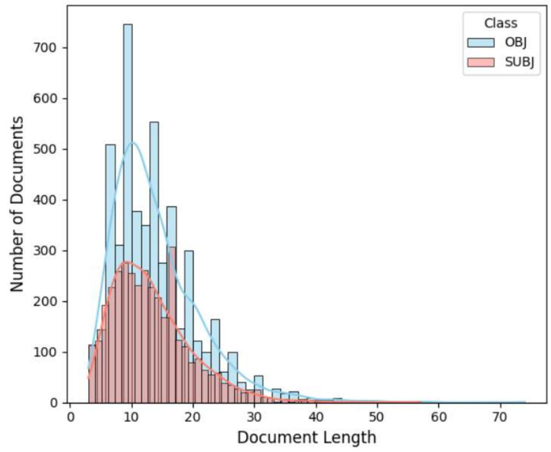
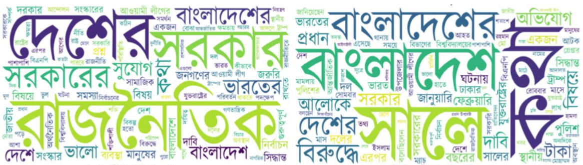


Figure 2: Distribution of sentence length (in words) for *SUBJ* and *OBJ* classes

To further explore the vocabulary and writing patterns, word clouds were generated separately for the subjective and objective classes. Figure 3 shows word clouds of the top 100 words from the dataset. Figure 3 revealed that the subjective class



(a) SUBJ class **(b) OBJ class**

Figure 3: Word clouds of the top 100 words from the dataset.

frequently includes words such as রাজনৈতিক (political), ভাল (good), and জরুরি (urgent), etc., which reflect opinions, emotions, or evaluations. In contrast, বাংলাদেশ (Bangladesh), কথা (talk), and সালে (in the year), etc., focus more on reporting events and presenting information in objective samples.

4 Methodology

This section describes our POS-aware transformer approach to subjectivity detection, integrating syntactic cues with contextual representations. We also outline the machine learning, deep learning, and transformer baselines evaluated for comparison.

4.1 Baselines

To evaluate the effectiveness of the proposed approach, we also implemented a range of ML, DL, and transformer-based baselines. All models are trained and assessed on the developed **BeNSD** dataset. Preprocessing, tokenization, and feature extraction techniques are applied uniformly across models where applicable.

- **ML Baselines:** This study examines several traditional ML models, including Logistic Regression (LR), Support Vector Machine (SVM), Naïve Bayes (NB), Random Forest (RF), XGBoost, and their ensembles. The text data is first converted to lowercase, and stopwords are removed. Next, the TF-IDF and CountVectorizer methods are used to convert the text into numerical features. These features are then used to train each model, with hyperparameters adjusted based on the validation results. Key hyperparameters include 100 estimators for RF, *logloss* evaluation metric for XGBoost, and a maximum of 1000 iterations for Logistic Regression.

- **DL Baselines:** To improve the quality of text representation, we explore DL models with pre-trained word embeddings. We experiment with four architectures: CNN, LSTM, BiLSTM, and their hybrid combinations. All models start with an embedding layer initialized with pre-trained Bengali embeddings from Word2Vec (Goldberg and Levy, 2014), GloVe (Pennington et al., 2014), or FastText (Bojanowski et al., 2016). We tune key hyperparameters, such as learning rate, batch size, and sequence length, using validation data, and apply dropout and early stopping to reduce overfitting. All models are trained for up to 20 epochs, with early stopping saving the best model based on the validation loss. The LSTM model with 300-dimensional Word2Vec embeddings, a learning rate of 0.001, a batch size of 64, and the Adam optimizer performs best, achieving optimal results around 7 epochs. Table 3 illustrates the tuned hyperparameters for DL models. Hyperparameters are tuned using a mix of grid and random search across predefined ranges. We fine-tune key hyperparameters, such as sequence length, LSTM units, batch size, optimizer, and learning rate, based on validation performance. To reduce overfitting, we employ dropout and L2 regularization techniques.

Hyperparameter	**Search Space**	**L**	**B**	**C + B**
Vocabulary Size	[5000, 10000, 15000]	10000	10000	10000
Sequence Length	[64, 128, 256]	128	128	128
Embed. Dim	[100, 300]	300	300	300
LSTM Units	[64, 128, 256]	128	64	64
Dense Units	[32, 64, 128]	64	32	32
Batch Size	[32, 64, 128]	64	64	64
Learning Rate	[0.001, 0.01, 0.0001]	0.001	0.001	0.001
Optimizer	[adam, rmsprop]	adam	adam	adam
Conv Filters	[64, 128, 256]	–	–	64
Conv Kernel Size	[3, 5, 7]	–	–	3

Table 3: Hyperparameter summary of DL models. L, C, and B denote LSTM, CNN, and BiLSTM methods, respectively.

- **Transformer Baselines:** This work examines seven pre-trained transformer baselines and evaluates their performance on the dataset developed for subjectivity detection in Bengali news articles. The models are: Bangla-BERT-1 (Sarker, 2020), Bangla-BERT-2 (Bhattacharjee et al., 2021), Multilingual Bidirectional Encoder Representations from Transformers (m-BERT) (Devlin et al., 2018), distilled version of BERT (distilBERT) (Sanh et al., 2019) , cross-lingual version of robustly Optimized BERT (XLM-Roberta) (Conneau et al., 2020), Multilingual Representations for Indian Languages (MuRIL) (Khanuja et al., 2021) and indicBERT (Kakwani et al., 2020).

  All models are available in the Hugging Face Transformers library[2]. The m-BERT model has 12 layers, 12 attention heads, and 110 million parameters. We also evaluate distilbert-base-multilingual-cased, which has 6 layers and 768 hidden dimensions. MuRIL is trained on 17 Indian languages, including Bengali, using both monolingual and transliterated data. It provides improved contextual understanding for Indic-language tasks and is part of our evaluation. Bangla-BERT-1 is trained on the Bengali Common Crawl corpus using the base BERT architecture. We also include Bangla BERT-2, which is trained on a larger Bengali corpus and optimized for NLP tasks in Bengali. We chose XLM-RoBERTa for its strong multilingual performance. It is trained on 100 languages and has 12 transformer layers with 125 million parameters. IndicBERT is a lightweight model for Indic languages, including Bengali. We fine-tune all models on our dataset with various hyperparameter settings. We tune key hyperparameters, such as batch size, learning rate, weight decay, and the learning rate scheduler (linear with warmup), among others. Models are trained for up to 15 epochs with early stopping and the Adam optimizer. A learning rate of 1e-5 with weight decay of 0.01 is used. Evaluation and model checkpointing are performed every 300 steps based on the F1 score. Mixed-precision training (fp16) is enabled to accelerate training.

4.2 Proposed Architecture

Combining syntactic and lexical features with transformer models improves downstream task performance (Shi et al., 2022). Motivated by these findings, this work presents a POS-aware classification method that merges POS features with pre-trained contextual embeddings. The proposed model leverages Bengali POS tags and semantic information to enhance the identification of linguistic subjective patterns, providing a richer context for classification. Figure 4 illustrates the architecture of the proposed model.

4.2.1 Contextual and POS Embedding Extraction

For each input sentence, $S = [w_1, w_2, w_3, \ldots, w_m]$, the tokenizer produces a subword token sequence, $\text{TOK}(S) = [x_1, x_2, x_3, \ldots, x_n]$. Each token x_i is converted into an embedding vector, $\mathbf{E}^x = (\mathbf{e}_1^x, \mathbf{e}_2^x, \mathbf{e}_3^x, \ldots, \mathbf{e}_n^x)$, where $\mathbf{e}_i^x \in \mathbb{R}^{768}$ using the embedding layer. It combines token embedding $\mathbf{E}_{\text{token}}(x_i)$, positional embedding

[2] https://huggingface.co

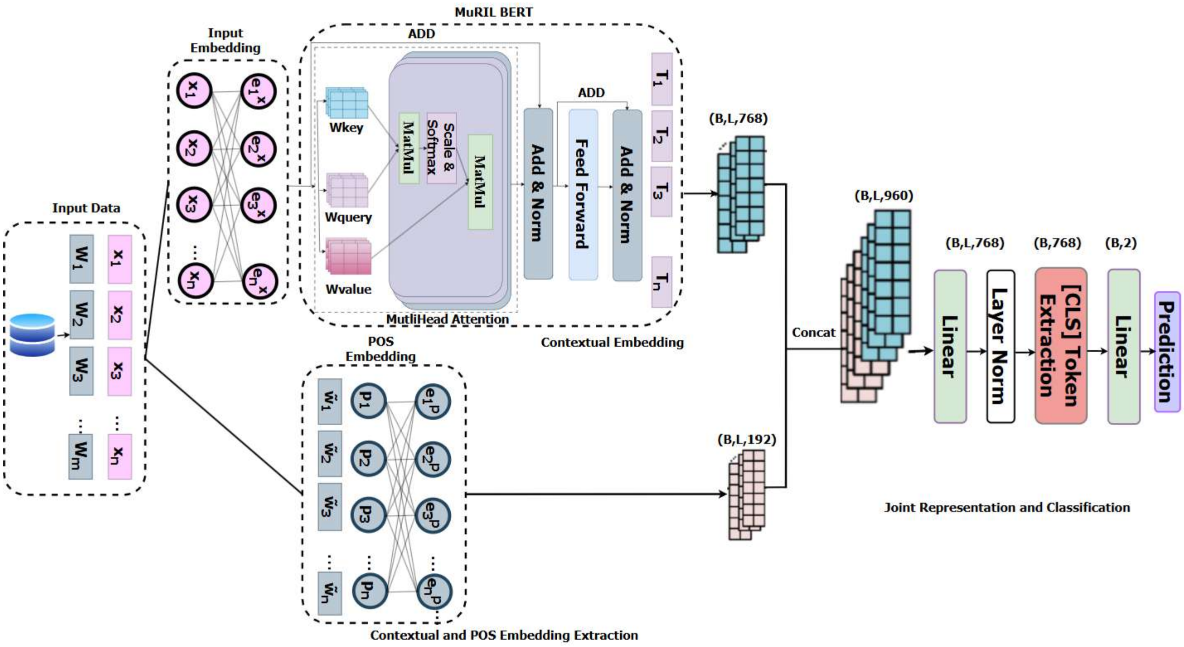


Figure 4: Architecture of the proposed model, where B: batch size, L: sequence length, x_i: subword tokens, $\tilde{w}_i$: original words mapped from tokens, and p_i: POS tag.

$\mathbf{E}_{\text{pos}}(i)$, and segment embedding $\mathbf{E}_{\text{segment}}(s)$. To obtain contextual embeddings, these input embeddings are fed into MuRIL's transformer layers as shown in Eq. 1.

$$\begin{aligned} T_i &= \text{MuRIL}(\mathbf{e}_i^x) \\ &= \text{MuRIL}\big(\mathbf{E}_{\text{token}}(x_i) + \mathbf{E}_{\text{pos}}(i) + \mathbf{E}_{\text{segment}}(s)\big) \end{aligned} \tag{1}$$

It produces contextual embedding, $\mathbf{T} = [\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n]$, $\mathbf{T}_i \in \mathbb{R}^{768}$ that captures semantic information.

For POS embedding, since tokenization may split a word into multiple subword tokens, we assign the POS tag of the original word to all its tokens. This ensures that the model receives the correct syntactic information aligned with its input tokens. For each token in the tokenized sequence, we track the original word it came from, denoted as $\tilde{w}_i$, and its POS tag, denoted as p_i. The BNLP library [3] is used for extracting POS tags. We map each POS tag p_i to an integer ID, denoted as $\text{id}(p_i)$, over a set of 14 predefined coarse-grained categories. These categories include core syntactic classes such as nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, interjections, determiners, and punctuation, as well as special tokens like PAD, UNK, CLS, and SEP. We then create trainable POS embeddings of 64 dimensions as Eq. 2.

$$\mathbf{p}_i = \text{Embed}_{pos}[\text{id}(p_i)], \quad \mathbf{p}_i \in \mathbb{R}^{64} \tag{2}$$

To enrich the representation and improve the models ability to capture syntactic nuances, we project the POS embeddings into a higher-dimensional space, as shown in Eq. 3.

$$\tilde{\mathbf{p}}_i = \mathbf{W}_{proj}\mathbf{p}_i + \mathbf{b}_{proj}, \quad \tilde{\mathbf{p}}_i \in \mathbb{R}^{192} \tag{3}$$

where $\mathbf{W}_{proj} \in \mathbb{R}^{192\times 64}$ is a trainable projection matrix and $\mathbf{b}_{proj} \in \mathbb{R}^{192}$ is a trainable bias vector. This projection enhances the performance of POS embeddings, making syntactic information more meaningful for the model.

4.2.2 Joint Representation and Classification

After projecting 64-dimensional POS embeddings to a 192-dimensional representation, for each token, we concatenate the contextual embedding $\mathbf{T}_i \in \mathbb{R}^{768}$ from MuRIL with the projected POS embedding $\tilde{\mathbf{p}}_i \in \mathbb{R}^{192}$ to form a fused representation as shown in Eq. 4.

$$\mathbf{z}_i = \begin{bmatrix} \mathbf{T}_i \\ \tilde{\mathbf{p}}_i \end{bmatrix}, \quad \mathbf{z}_i \in \mathbb{R}^{960} \tag{4}$$

This combined 960-dimensional representation, $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n]$, captures both the syntactic

[3] https://github.com/sagorbrur/bnlp

and contextual aspects of each token. Each fused vector $\mathbf{z}_i$ is then passed through a feed-forward fusion layer with GELU activation function, which is chosen for its smooth non-linearity and effectiveness with small input values. After that, to stabilize the training and accelerate the convergence, we use layer normalization, which produces the transformed representation, $\mathbf{u}_i \in \mathbb{R}^{768}$ as defined in Eq. 5.

$$\mathbf{u}_i = \text{LayerNorm}\Big(\text{GELU}\big(\mathbf{W}_{\text{fusion}}\mathbf{z}_i + \mathbf{b}_{\text{fusion}}\big)\Big) \tag{5}$$

We train with a learning rate of 1×10^{-5}, weight decay of 0.01, and batch size of 16. A linear scheduler with a 0.1 warm-up ratio and early stopping with a patience of 10 epochs is applied. Finally, to perform classification, we extract the [CLS] token, which serves as the sentence-level representation. It is passed through a final dense layer to predict subjectivity. The whole prediction function can be expressed as Eq. 6.

$$\hat{y} = \arg\max\big(\text{softmax}(\mathbf{W}_{\text{cls}}\,\mathbf{u}_{\text{[CLS]}} + \mathbf{b}_{\text{cls}})\big) \tag{6}$$

5 Experiments

All experiments are conducted on the Kaggle platform using Python 3. Notebooks are executed in a GPU-enabled Kaggle environment with NVIDIA Tesla P100 GPUs and 16GB of RAM. For data manipulation, we use pandas (2.2.3) and numpy (1.26.4). Traditional machine learning models are implemented using scikit-learn (1.2.2). In contrast, deep learning models are built with Keras (3.8.0) and TensorFlow (2.18.0). For transformer-based models, we utilize PyTorch (version 2.6.0) and the Hugging Face Transformers library.

5.1 Results and Discussions

While the macro F1-score (F1) is used to assess model performance, other standard metrics such as accuracy (A), precision (P), and recall (R) are also reported. Table 4 shows how different baseline models performed on subjectivity detection in Bengali on **BeNSD** dataset.

Performance of ML and DL Models: Traditional ML approaches are less effective for Bengali subjectivity detection. SVM with TF-IDF achieves the highest mF1 among ML models (73.06%). In contrast, DL models improve performance: the best, LSTM with Word2Vec, achieves an F1 of 83.49%an approximate 10-point gain. This highlights the importance of distributed representations and sequential modeling in detecting subjectivity in Bengali text.

Superior performance of Transformer models: Transformer-based models consistently outperform both traditional ML and DL approaches. It shows performance ranging from 80.13% (IndicBERT) to 92.09% (MuRIL). Notably, multilingual models like mBERT (87.20%) and XLM-R (90.33%), as well as Bengali-specific models such as Bangla-BERT variants (88.05-91.64%), demonstrate superior performance. These results suggest that both multilingual knowledge and language-specific contextual information play a crucial role in enhancing the detection of subjectivity in Bengali texts..

Impact of POS Integration Varies by Model Architecture: The integration of POS embedding shows varying degrees of improvement across different transformer models. Lower-performing models, such as IndicBERT, gain substantially from POS integration (+2.40 F1 scores), while some mid-tier models, like mDistilBERT, show modest improvements (+0.64 F1 scores). However, the relationship is not uniform across all high-performing models. Bangla-BERT-1 shows minimal gain (+0.01 F1), but other strong performers achieve more significant benefits.

POS-Aware Technique Enhances Model's Performance: The proposed POS-Aware-MuRIL achieves the highest performance with a F1 score of 93.35%. This represents a notable 1.26-point improvement over the base MuRIL. The improvement results from both the integration of POS embedding with the transformer's contextual embedding and an effective architecture. This architecture employs GELU activation and layer normalization to process combined features. These techniques enable the model to capture both contextual and syntactic information, thereby helping identify features indicative of subjectivity in Bengali news articles. Moreover, consistent improvements in recall across all POS-enhanced models (ranging from +0.47 to +1.90) suggest that syntactic features help identify subtle subjective patterns that purely contextual models might miss. Appendix A presents an ablation study to analyze the impact of POS embedding on performance.

Impact of Fusion Strategy: We evaluate five fusion mechanisms for integrating POS embeddings with MuRIL BERT representations. Table 5 illustrates the results of fusion techniques. Among

ML Models				
Classifier	**A (%)**	**P(%)**	**R(%)**	**F1(%)**
XGBoost (CountVec)	71.59	70.17	56.71	62.73
RF (TF-IDF)	72.63	67.68	67.12	67.40
RF (CountVec)	73.21	67.92	69.04	68.48
NB (TF-IDF)	76.10	78.42	59.73	67.81
LR (TF-IDF)	76.21	76.07	63.56	69.25
SVM (CountVec)	75.17	70.27	71.23	70.75
NB (CountVec)	75.52	71.19	70.41	70.80
LR (CountVec)	76.21	73.18	68.77	70.90
Ensemble (CountVec)	76.91	74.92	67.95	71.26
SVM (TF-IDF)	77.60	74.08	72.05	**73.06**
DL Models				
Classifier	**A(%)**	**P(%)**	**R(%)**	**F1(%)**
BiLSTM (GloVe)	79.45	70.19	79.18	78.50
CNN+BiLSTM (GloVe)	81.87	75.62	84.11	79.64
CNN+BiLSTM (FastText)	81.76	74.01	87.40	80.15
LSTM (GloVe)	82.79	76.34	85.75	80.77
LSTM (FastText)	84.64	81.52	82.19	81.86
BiLSTM (FastText)	84.99	82.19	82.20	82.19
BiLSTM (Word2Vec)	84.76	79.80	85.48	82.54
CNN+BiLSTM (Word2Vec)	84.87	80.31	84.93	82.56
LSTM (Word2Vec)	85.33	79.46	87.95	**83.49**
Transformers				
Classifier	**A(%)**	**P(%)**	**R(%)**	**F1(%)**
IndicBERT	80.48	79.99	80.38	80.13
+POS	82.79	82.35	82.90	82.53
Δ	+2.31	+2.36	+2.52	+2.40
mDistilBERT	84.53	84.22	83.95	84.07
+POS	85.22	85.10	84.44	84.71
Δ	+0.69	+0.88	+0.49	+0.64
mBERT	87.53	87.60	87.04	87.20
+POS	87.99	87.69	87.68	87.69
Δ	+0.46	+0.09	+0.64	+0.49
Bangla-BERT-1	88.22	87.82	88.12	88.05
+POS	88.22	87.85	88.59	88.06
Δ	+0.00	+0.05	+0.47	+0.01
XLM-Roberta	90.65	90.75	90.02	90.33
+POS	91.57	91.49	91.19	91.33
Δ	+0.92	+0.74	+1.17	+1.00
Bangla-BERT-2	91.80	92.02	92.03	91.64
+POS	92.38	92.03	92.52	92.23
Δ	+0.58	+0.36	+0.49	+0.59
MuRIL	92.38	92.79	91.63	92.09
+POS-Aware (Proposed)	93.42	93.10	93.53	**93.35**
Δ	+1.04	+0.31	+1.90	+1.26

Table 4: Performance of various models on the subjectivity detection task

them, concatenation achieves the highest macro F1 score of 93.35%, followed by gated fusion at 92.49%. The proposed concatenation-based fusion outperforms the gated fusion by 0.86% and the additive fusion by 0.91%. All fusion strategies demonstrate strong performance; however, the simple concatenation-based fusion shows the most robust overall performance when integrating both embeddings. Element-wise strategy, such as additive and multiplicative fusion, outperforms attention-based fusion. The strong performance of the concatenation-based fusion in our POS-Aware MuRIL model shows that keeping features separate helps the model use linguistic information more effectively.

Fusion Type	**Ac (%)**	**Pr (%)**	**Re (%)**	**F1 (%)**
Attention	92.26	92.82	91.42	91.95
Multiplicative	92.61	92.48	92.35	92.41
Additive	92.61	92.35	92.53	92.44
Gated	92.73	92.83	92.23	92.49
Concatenation	93.42	93.10	93.53	**93.35**

Table 5: Performance comparison of different fusion strategies

5.2 Comparison with Existing Approaches

To the best of our knowledge, no publicly available dataset exists for subjectivity detection in Bengali news articles. To facilitate comparison in this context, we implement and adapt several existing techniques from similar domains (Antici et al., 2023; Sagnika et al., 2021; Paran et al., 2024; Dey et al., 2023) to the **BeNSD** dataset, ensuring consistency across these approaches. Table 6 presents a comparative analysis of F1-scores. Notably, the proposed *POS-Aware MuRIL* model achieves the highest F1 score of 93.35%, improving by 9.28% over LLaMA (Paran et al., 2024) and 2.02% over the XLM-R (Dey et al., 2023).

Approach	**mF1 (%)**
CNN-LSTM+Attention (Sagnika et al., 2021)	77.65
LLaMA-3-8B (Paran et al., 2024)	84.07
mBERT (Antici et al., 2023)	88.65
XLM-R (Dey et al., 2023)	91.33
POS-Aware MuRIL (Proposed)	**93.35**

Table 6: Comparison of existing approaches with the proposed method for Bengali subjectivity detection.

5.3 Error Analysis

The results confirmed that the proposed model detects subjectivity in Bengali news articles more effectively than the baselines. To better understand how the model performs, we carried out a detailed error analysis using both quantitative and qualitative methods.

Quantitative Error Analysis: Figure 5 shows the confusion matrix for the proposed *POS-Aware MuRIL* model on test sets. Of the 866 samples, 810 were correctly classified and 56 were misclassified. Among 365 subjective samples, 31 were misclassified, and among 501 objective samples, 25 were misclassified. This suggests the model misclassified both SUBJ and OBJ instances at nearly the same rate. The errors are not strongly biased toward one class but reflect challenges in capturing

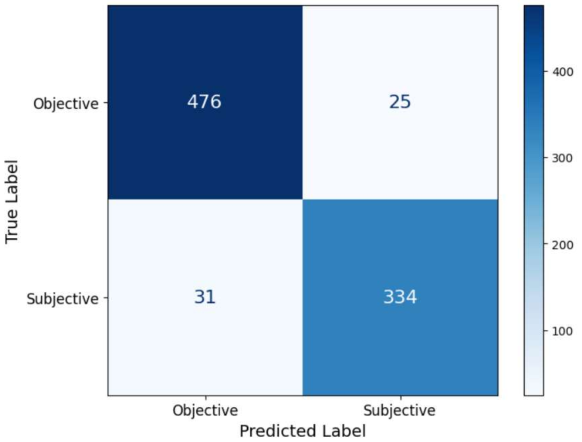


Figure 5: Confusion matrix for the proposed model on test data

nuanced distinctions. A primary reason is lexical ambiguity, in which certain words or phrases appear in both subject and object contexts depending on usage.

Figure 6 shows the 2D t-SNE plot, which indicates a clear separation between classes, with errors primarily located at the edges of the clusters, as highlighted by the rectangle. This suggests the model mainly struggles with ambiguous cases that fall between the two classes. These insights could help improve classification by adding more varied linguistic cues during training and refining feature representation.

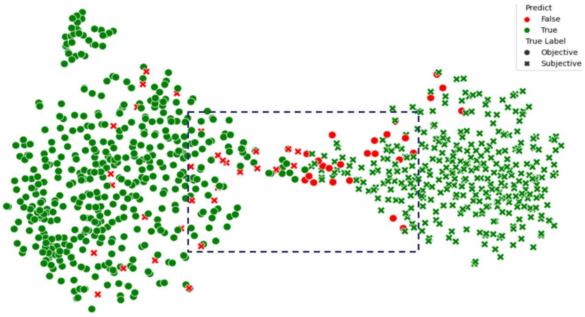


Figure 6: 2D t-SNE plot of *POS-Aware-MuRIL* embedding on test data.

Qualitative Error Analysis: Table 7 shows some sample texts that are correctly and incorrectly classified by the *POS-Aware MuRIL* model. The first and third samples were correctly classified, whereas the model failed to classify the second and fourth samples.

Text	**Ac**	**Pr**
বাংলাদেশের স্বার্থেই রোহিঙ্গা সংকটের সমাধান বা বর্তমানের চেয়ে ভালো বিকল্প বের করা উচিত (*For Bangladeshs own interest, the Rohingya crisis should be resolved or a better alternative than the current one should be found.*)	SUBJ	SUBJ
যেন বলা হলো—সন্ধ্যার পর ছাত্রীরা হল থেকে বের হয় বলেই এমন ঘটনা ঘটে (*As if to say, such incidents happen because female students leave the hall after dark.*)	SUBJ	OBJ
অল্টম্যানের পক্ষ থেকে তাদের চীনা প্রতিদ্বন্দ্বী ডিপসিকের প্রশংসা করে বলা হয়, এটা ভালো একটি মডেল (*On behalf of Altman, their Chinese rival DeepSeek was praised, saying its a good model.*)	OBJ	OBJ
ঐতিহ্যবাহী এই উৎসব দেখতে উৎসুক জনতার কোনো কমতি ছিল না (*There was no shortage of eager crowds to watch this traditional festival.*)	OBJ	SUBJ

Table 7: Some correctly and incorrectly classified samples by *POS-Aware MuRIL* model

The analysis of incorrect predictions reveals that it is challenging to identify texts with hidden subjectivity. These cases are tough to categorize because they do not clearly show subjective meaning. Sarcasm, a critical tone, and rhetorical questions, for example, often lead to misclassification. Additionally, the model incorrectly labels factual descriptions as subjective when they contain descriptive adjectives. Without context, it is challenging to classify these samples solely based on the text. Subjectivity typically relies on broader cues that sentence-level models often overlook. Using more context and a wider range of training data could help improve the models performance.

6 Conclusion

This work presented **BeNSD**, a manually annotated dataset for subjectivity detection in Bengali, containing 8,655 news articles from multiple online sources. Alongside, we introduce a transformer-based model (**POS-Aware-MuRIL**) that fuses MuRIL's contextual embeddings with POS embeddings to boost classification accuracy. Evaluation shows the proposed model surpasses machine learning, deep learning, and transformer baselines, achieving the top macro F1 score (93.35%). Building on these results, we plan to expand the dataset, refine syntactic feature extraction through improved POS tagging, and investigate ensemble and multitask learning to further enhance system robustness and generalization.

Limitations

Although the proposed approach shows promising results, some limitations remain. First, because the POS tagger is not fine-tuned for this dataset, some syntactic information may be misidentified, leading to inaccurate feature extraction and misclassifications in the text. Additionally, frequent words that appear in both subjective and objective texts create class ambiguity, making it difficult for the model to distinguish between categories and reducing its ability to generalize. Moreover, context-specific language and sarcasm can cause the model to misinterpret meaning, further limiting classification accuracy. Addressing these issues by fine-tuning linguistic tools and expanding the dataset is crucial to improving performance. Furthermore, integrating large language models (LLMs) in future work could enhance the results.

Acknowledgements

This work is supported by the Directorate of Research and Extension (DRE) and NLP Lab, Chittagong University of Engineering & Technology (CUET), Chittagong 4349, Bangladesh.

Ethics Statement

The *BeNSD* dataset was created by a team that included a native Bengali-speaking NLP expert and three undergraduate students with NLP backgrounds. The team employed clear annotation guidelines that outlined data sources, collection steps, and task formats to ensure data consistency and accuracy. They also manually checked and corrected the data to ensure its accuracy. An NLP expert with more than 20 years of experience reviewed the final annotations to confirm their reliability. Throughout the process, the team adhered to ethical guidelines to ensure the data was fair and trustworthy.

References

Francesco Antici, Luca Bolognini, Matteo Antonio Inajetovic, Bogdan Ivasiuk, Andrea Galassi, and Federico Ruggeri. 2021. Subjectivita: An italian corpus for subjectivity detection in newspapers. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 12th International Conference of the CLEF Association, CLEF 2021, Virtual Event, September 21–24, 2021, Proceedings 12*, pages 40–52. Springer.

Francesco Antici, Andrea Galassi, Federico Ruggeri, Katerina Korre, Arianna Muti, Alessandra Bardi, Alice Fedotova, and Alberto Barrón-Cedeño. 2023. A corpus for sentence-level subjectivity detection on english news articles. *arXiv preprint arXiv:2305.18034*.

Abhik Bhattacharjee, Tahmid Hasan, Kazi Samin, M Sohel Rahman, Anindya Iqbal, and Rifat Shahriyar. 2021. Banglabert: Combating embedding barrier for low-resource language understanding. *arXiv preprint arXiv:2101.00204*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information.

Iti Chaturvedi, Edoardo Ragusa, Paolo Gastaldo, Rodolfo Zunino, and Erik Cambria. 2017. Bayesian network based extreme learning machine for subjectivity detection. *Journal of the Franklin Institute*, 355.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

K. Dey, P. Tarannum, M. A. Hasan, and S. R. H. Noori. 2023. Nn at checkthat!-2023: Subjectivity in news articles classification with transformer based models. In *Working Notes of CLEF 2023 Conference and Labs of the Evaluation Forum*, volume 3497 of *CEUR Workshop Proceedings*.

Satyam Dwivedi and Sanjukta Ghosh. 2022. Subjectivity identification through lexical rules. *SN Computer Science*, 3(1):32.

Satyam Dwivedi, Sanjukta Ghosh, and Shivam Dwivedi. 2024. Navigating linguistic diversity: In-context learning and prompt engineering for subjectivity analysis in low-resource languages. *SN Comput. Sci.*, 5(4).

Raphael Antonius Frick. 2023. Fraunhofer sit at checkthat!-2023: Can llms be used for data augmentation & few-shot classification? detecting subjectivity in text using chatgpt. In *CLEF (Working Notes)*, pages 329–336.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*.

Simran Khanuja, Sumanth Doddapaneni, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali. 2021. Muril: Multilingual representations for indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 607–622.

Georgi Pachov, Dimitar Dimitrov, Ivan Koychev, and Preslav Nakov. 2023. Gpachov at checkthat! 2023: A diverse multi-approach ensemble for subjectivity detection in news articles. *Preprint*, arXiv:2309.06844.

Ashraful Paran, Md Hossain, Symom Shohan, Jawad Hossain, Shawly Ahsan, and Moshiul Hoque. 2024. Semanticcuetsync at checkthat! 2024: Finding subjectivity in news articles using llama notebook for the checkthat! lab at clef 2024.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Santwana Sagnika, B. Mishra, and Saroj Meher. 2021. An attention-based cnn-lstm model for subjectivity detection in opinion-mining. *Neural Computing and Applications*, 33:1–14.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Sagor Sarker. 2020. Banglabert: Bengali mask language model for bengali language understading.

Ranjan Satapathy, Shweta Pardeshi, and Erik Cambria. 2022. Polarity and subjectivity detection with multitask learning and bert embedding. *Preprint*, arXiv:2201.05363.

Yu Shi, Xi Zhang, and Ning Yu. 2022. Pl-transformer: a pos-aware and layer ensemble transformer for text classification. *Neural Computing and Applications*, 35.

R. Suwaileh, M. Hasanain, F. Hubail, W. Zaghouani, and F. Alam. 2025. Thatiar: Subjectivity detection in arabic news sentences. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 19, pages 2587–2602.

A Ablation Study

To identify the best configuration for tasks, we conduct an ablation analysis of key factors, including the impact of POS embedding dimensions for integrating POS embeddings with MuRIL BERT. As POS embeddings play a pivotal role in performance, we vary the POS embedding dimension from 16 to 512 to determine the optimal size for capturing syntactic information. Figure 7 shows that dimensions between 32 and 128 yield stable performance, with 64 achieving the best F1 score of 0.9335. Smaller dimensions fail to capture a sufficient number of syntactic patterns. On the other hand, larger dimensions add unnecessary parameters without improving performance. This indicates that moderate embedding sizes offer a better trade-off between expressiveness and efficiency.

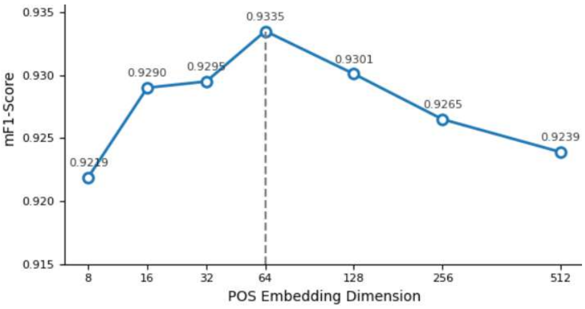


Figure 7: Impact of POS embedding dimension on model performance

Gen-mABSA-T5: A Multilingual Zero-Shot Generative Framework for Aspect-Based Sentiment Analysis

Shabrina Akter Shahana
Dept. of CSE, Jahangirnagar University
shabrina.stu2018@juniv.edu

Nuzhat Nairy Afrin
Dept. of CSE, Jahangirnagar University
nuzhat.stu2019@juniv.edu

Md. Musfique Anwar
Dept. of CSE, Jahangirnagar University
manwar@juniv.edu

Israt Jahan
Dept. of CSE, Jahangirnagar University
israt@juniv.edu

Abstract

Aspect-Based Sentiment Analysis (ABSA) identifies sentiments toward specific aspects of an entity. While progress has been substantial for high-resource languages such as English, low-resource languages like Bangla remain underexplored due to the limited availability of annotated data and linguistic challenges. We propose Gen-mABSA-T5, a multilingual zero-shot generative framework for ABSA based on Flan-T5, incorporating prompt engineering and Natural Language Inference (NLI). Without task-specific training, Gen-mABSA-T5 achieves state-of-the-art zero-shot accuracy of 61.56% on the Large Bangla Corpus, 73.50% on SemEval Laptop, and 73.56% on SemEval Restaurant outperforms both English and Bangla task-specific models in the zero-shot settings. It delivers reasonable performance against very large general-purpose models on both English and Bangla benchmarks. These results highlight the effectiveness of generative, zero-shot approaches for ABSA in low-resource and multilingual settings.

1 Introduction

Bangla is the seventh most spoken language worldwide (Shammi et al., 2023), yet Aspect-Based Sentiment Analysis (ABSA) for Bangla remains underexplored due to scarce annotated data and language-specific complexities. ABSA, a fine-grained extension of sentiment analysis, identifies sentiments toward specific aspects of an entity rather than providing only an overall polarity. For example, in "The phone's camera is excellent, but the battery is weak," the sentiment is positive for the *camera* but negative for the *battery*. This level of detail makes ABSA valuable in applications such as e-commerce, service reviews, and social media monitoring, where organizations require targeted insights into user opinions.

Despite its importance, ABSA faces significant challenges. Traditional supervised approaches rely on large domain-specific datasets, which are rarely available for low-resource languages like Bangla. Even multilingual transformer models such as BERT and RoBERTa, though effective in English, underperform in Bangla due to limited exposure to its morphology and syntax. Furthermore, ABSA is complicated by sentiment ambiguity and domain variation, highlighting the need for approaches that minimize reliance on annotated data while generalizing across domains and languages.

Zero-shot ABSA addresses these issues by enabling models to identify aspects and classify their sentiment without task-specific training. Instead of requiring labeled corpora for each new domain, zero-shot models leverage pre-trained knowledge and adapt through mechanisms such as prompt engineering and natural language inference (NLI). This capability is particularly valuable for Bangla, where annotated resources are scarce, and for multilingual contexts, where scalability is essential.

This work introduces Gen-mABSA-T5, a multilingual generative zero-shot framework for ABSA built on Flan-T5. By integrating prompt engineering and with the framework's NLI capabilities, Gen-mABSA-T5 performs aspect sentiment classification in Bangla and English without domain-specific supervision. Evaluations in SemEval 2014 (English) (Pontiki et al., 2014) and the Large Bangla Corpus demonstrate that Gen-mABSA-T5 achieves state-of-the-art zero-shot accuracy, outperforming both English and Bangla task-specific models in zero-shot settings. It delivers reasonable performance against very large general-purpose models on both English and Bangla benchmarks, underscoring the model's adaptability to both high and low-resource settings.

The contributions of this work are as follows:

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 142–150
December 23, 2025 ©2025 Association for Computational Linguistics

- We propose Gen-mABSA-T5, a multilingual zero-shot generative framework for ABSA that does not require task-specific training data.

- We conducted an extensive evaluation on English and Bangla datasets, achieving state-of-the-art zero-shot results for Bangla and reasonable zero-shot performance for English.

- We demonstrate the effectiveness of prompt engineering and NLI for cross-lingual ABSA, highlighting the scalability of zero-shot generative approaches for low-resource languages.

2 Literature Review

The field of Aspect-Based Sentiment Analysis (ABSA) has witnessed significant advances in recent years, driven by developments in natural language processing and deep learning. Existing research covers a broad spectrum, from rule-based and feature-based approaches to state-of-the-art transformer models. However, most prior work relies heavily on large annotated datasets, which limits applicability in low-resource languages. Recent trends focus on multilingual and zero-shot approaches to overcome these limitations. We reviewed relevant studies, identifying strengths, limitations, and gaps that motivate the proposed Gen-mABSA-T5 framework.

SA has evolved from lexicon and rule-based systems to statistical models, deep learning, and transfer learning. In English, early approaches used sentiment lexicons (e.g., SentiWordNet) and parsers, with SemEval(2014) establishing benchmarks. The advent of BERT and related transformers marked a paradigm shift, while zero-shot and few-shot learning extended applicability in low-resource settings (Li and Xiang., 2020; Pathan and Prakash., 2022).

Bangla SA has developed more slowly due to limited resources. Early studies relied on translated or code-mixed datasets (Dey and Sarker., 2019; Mahtab et al., 2018), lexicon-based methods (Rabeya and Sattar., 2022), and multilingual models (Alam and Kamal., 2024). More recent work has applied deep learning and multilingual transformers (Agüero-Torales and López-Herrera., 2021), although high-quality annotated corpora remain scarce.

ABSA refines SA by assigning sentiment at the aspect level. Early English ABSA employed lexicon-based methods and dependency parsing, later replaced by pre-trained models, unified pipelines, and generative frameworks such as T5 and BART (Zhang and Lam., 2021; Huan and Guo., 2022). In Bangla, research began with traditional ML baselines (Rahman and Dey., Data 3, no. 2 (2018; Rahman and Kumar Dey, 2018), progressed to deep learning with CNN, LSTM, and RNN architectures (Mahfuz Ahmed Masum and Islam., 2020; Md Morshedul Islam and Mynoddin., 2023), and has recently adopted LLMs (Shihab Ahmed and Mridha, 2024; Md Akash Rahman and Andersson., 2024; Junayed Hossain and Monir., 2024), improving performance on complex tasks such as implicit aspect detection.

Zero-shot ABSA has emerged as a promising alternative to supervised approaches. In English, defining ABSA as a Natural Language Inference (NLI) task enabled models such as BERT and T5 to perform aspect extraction and sentiment classification without domain-specific training (Shu et al., 2022). More recently, GPT-3 and GPT-4 have demonstrated robust zero-shot performance across unseen domains. For Bangla, progress is limited by morphological complexity and resource scarcity, though multilingual models (mBERT, XLM-R) enable some transfer from English with mixed results.

Comparing approaches, traditional ML offers interpretability on small datasets, deep learning improves accuracy but requires large corpora, while LLMs provide advanced contextual understanding and strong performance in multilingual and zero-shot settings.

In summary, English SA and ABSA have matured through robust datasets and methodologies, while Bangla research remains constrained by limited resources. Recent advances in deep learning and LLMs, however, have opened promising directions for Bangla sentiment analysis and ABSA. These developments underscore the importance of scalable, zero-shot approaches capable of bridging the gap between high- and low-resource languages.

3 Proposed Methodology: Gen-mABSA-T5

We introduce Gen-mABSA-T5, a multilingual zero-shot generative framework for aspect-based sentiment analysis (ABSA), which addresses challenges in low-resource languages such as Bengali, where annotated data is limited. Unlike traditional ABSA models that depend on large labeled datasets, Gen-mABSA-T5 leverages zero-shot learning, prompt engineering, and a generative transformer architecture to perform ABSA without task-specific training.

3.1 System Overview

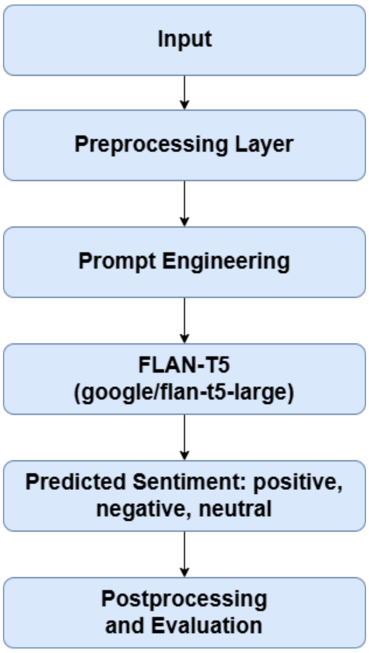

Input
Preprocessing Layer
Prompt Engineering
FLAN-T5
(google/flan-t5-large)
Predicted Sentiment: positive, negative, neutral
Postprocessing and Evaluation

Figure 1: Gen-mABSA-T5 System Overview

The framework is based on Flan-T5 and integrates prompt engineering with NLI to classify sentiment toward a given aspect in a fully generative manner. Input consists of a sentence and an explicitly provided aspect; output is a single sentiment label (positive, negative, or neutral). No fine-tuning or separate NLI head is used. NLI is achieved through carefully designed prompts that take advantage of Flan-T5's pre-training on inference tasks.

3.2 Model Architecture

Figure 2 shows the full end-to-end pipeline of Gen-mABSA-T5. The model receives a raw sentence together with an explicitly annotated aspect term (no aspect extraction is performed). The sentence first passes through a preprocessing layer that removes all emojis, deletes punctuation marks used in both English and Bangla (including the Bangla full stop , double full stop , curly quotes , and zero-width joiners that can break compound Bangla words), and collapses multiple spaces into one. This cleaned sentence is inserted into a prompt style: For example, in English, "Sentence: [text]. Aspect: [aspect]. Classify sentiment toward the aspect: positive, negative, or neutral." and in Bangla the exact Bangla translation of the same instruction. The completed prompt is fed directly to the publicly available `Google/Flan-t5-large` model under zero-shot inference with fixed settings (temperature 0, maximum 10 new tokens). The model generates the word "positive", "negative", or "neutral", which is then converted to the integer labels used for evaluation (positive = 2, negative = 0, neutral = 1).

3.3 Zero-Shot and NLI Framing

Zero-shot ABSA addresses these issues by enabling models to identify aspects and classify their sentiment without task-specific training. Instead of requiring labeled corpora for each new domain, zero-shot models leverage pre-trained knowledge and adapt through mechanisms such as prompt engineering and NLI. This capability is particularly valuable for Bangla, where annotated resources are scarce, and for multilingual contexts, where scalability is essential.

Gen-mABSA-T5 performs aspect sentiment classification in Bangla and English without domain-specific supervision. The framework frames ABSA as an NLI task within a generative paradigm: the input sentence serves as the premise, while a hypothesis template (The sentiment toward [aspect] is [positive/negative/neutral]) is constructed via a prompt. Flan-T5 then generates a prediction, which is directly mapped to positive, negative, or neutral. No separate NLI head or classifier is used. The reasoning of NLI emerges from Flan-T5's pre-training on datasets such as MNLI and SNLI, activated only through structured instructions. This design ensures full zero-shot operation with no fine-tuning.

3.4 Prompt Engineering and Ablation Studies

By integrating prompt engineering and NLI, Gen-mABSA-T5 performs aspect sentiment classification in Bangla and English without domain-specific supervision. Prompts were iteratively designed using a held-out validation split (10% of each training set). We tested 10 English and 10 Bangla variants, including concise, NLI-style, structured output, etc. formats. The best-

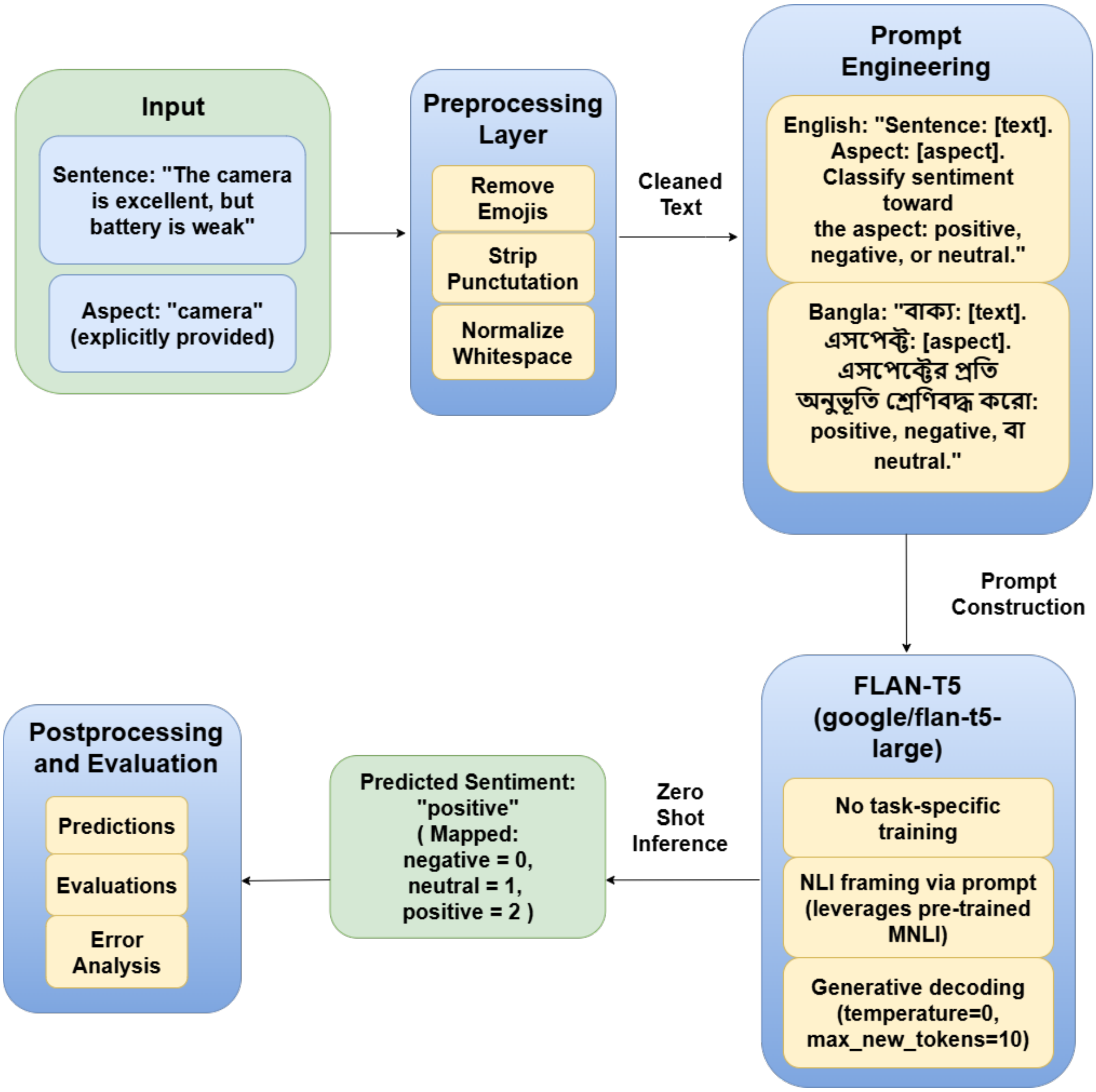

Input
Sentence: "The camera is excellent, but battery is weak"
Aspect: "camera" (explicitly provided)
Preprocessing Layer
Remove Emojis
Strip Punctutation
Normalize Whitespace
Cleaned Text
Prompt Engineering
English: "Sentence: [text]. Aspect: [aspect]. Classify sentiment toward the aspect: positive, negative, or neutral."
Bangla: "বাক্য: [text]. এসপেক্ট: [aspect]. এসপেক্টের প্রতি অনুভূতি শ্রেণিবদ্ধ করো: positive, negative, বা neutral."
Prompt Construction
FLAN-T5 (google/flan-t5-large)
No task-specific training
NLI framing via prompt (leverages pre-trained MNLI)
Generative decoding (temperature=0, max_new_tokens=10)
Zero Shot Inference
Predicted Sentiment: "positive" (Mapped: negative = 0, neutral = 1, positive = 2)
Postprocessing and Evaluation
Predictions
Evaluations
Error Analysis

Figure 2: End-to-end pipeline of Gen-mABSA-T5

performing prompts are the following concise and explicit prompts:

> You are a sentiment analysis expert.
> Sentence: [sentence].
> What is the sentiment towards [aspect]?
> Choose only one: positive, negative, or neutral.

For Bengali, prompts are adapted to its morphology and syntax:

> তুমি একজন অনুভূতি বিশ্লেষণ বিশেষজ্ঞ।
> বাক্য: [sentence]
> [aspect]-এর প্রতি অনুভূতি কী?
> শুধুমাত্র একটি বেছে নাও: positive, negative, অথবা neutral।

Longer prompts and domain context reduced performance (-13.1% avg. accuracy), likely due to noise. Strict, concise instructions with explicit aspect focus yielded optimal zero-shot generalization. See Table 1.

3.5 Aspect Handling

The model assumes aspects are pre-provided from dataset annotations and does not perform automatic aspect extraction. Only explicit, single-word or short-phrase aspects are processed; implicit and multi-word aspects are out of scope. For each inference, one aspect is paired with the full sentence in the prompt. This design follows the standard evaluation protocol of SemEval 2014 (Pontiki et al., 2014) and the Bangla corpus, where

Prompt Style	SemEval 2014 Laptop	SemEval 2014 Restaurant	Bangla Corpus	Avg. Acc.
NLI-style Concise Explicit	0.735	0.736	0.616	0.696
Confidence Aware	0.710	0.728	0.625	0.688
Structured Output	0.716	0.721	0.612	0.683
Context Enhanced	0.619	0.594	0.508	0.574

Table 1: Ablation Study Evaluating the Impact of Different Prompt Designs on Gen-mABSA-T5's Accuracy.

aspect terms are given.

3.6 Implementation and Reproducibility

We implement Gen-mABSA-T5 using Google/Flan-t5-base (250M parameters) via Hugging Face Transformers. Inference is deterministic temperature=0.0, maximum new tokens is 10, batch size 32. The output is parsed by case-insensitive matching of positive, negative, or neutral. Text pre-processing removes emojis, normalizes punctuation, and collapses whitespace. Baselines (BERT-Multilingual, RoBERTa-Twitter, XLM-RoBERTa, Bangla-BERT) use sequence classification with 3 labels and weighted metrics. Zero-shot requires no training; few-shot samples 50 examples per aspect; fine-tuning runs 1500 steps (batch size 8, weight decay 0.01).

The SemEval 2014 Laptop and Restaurant (Pontiki et al., 2014) were split into train-test as the official distribution. The Large Bangla ABSA Corpus is split into an 80-20 ratio stratified by aspect.

The experiments were carried out on the Google NVIDIA T4 GPU (16 GB of VRAM) with 12 GB of RAM, using PyTorch and Python. All seeds were fixed at 42.

3.7 Contributions

Gen-mABSA-T5 advances ABSA by enabling multilingual support, including for low-resource languages, through zero-shot learning, reducing reliance on labeled data. Its generative approach enhances flexibility and performance in resource-limited settings, with applications in customer feedback and social media analysis. Gen-mABSA-T5 integrates NLI, multilingual transformers, and prompt engineering for scalable zero-shot ABSA.

4 Experimental Setup

We evaluated state-of-the-art sentiment analysis models for Aspect-Based Sentiment Analysis (ABSA) across diverse linguistic and domain contexts, focusing on zero-shot, few-shot, and fully fine-tuned settings. Evaluations use benchmark datasets: SemEval 2014 Task 4 (Restaurant and Laptop domains) for English, and the Large Bangla Corpus for Bangla. We assessed BERT-Multilingual, RoBERTa-Twitter, XLM-RoBERTa, Bangla-BERT, and our proposed Gen-mABSA-T5, a zero-shot multilingual ABSA model designed for low-resource languages like Bangla. The results demonstrate Gen-mABSA-T5's effectiveness, particularly in zero-shot scenarios, while identifying challenges and opportunities in multilingual ABSA.

4.1 Dataset

This section outlines the datasets used for evaluating Aspect-Based Sentiment Analysis (ABSA) models, covering English and Bangla to enable robust cross-linguistic and cross-domain analysis. For English, the SemEval 2014 Task 4 datasets, standard ABSA benchmarks, include the Restaurant domain (covering food, service, ambiance) and Laptop domain (e.g., battery life, screen quality), with sentences annotated for aspects and sentiment polarities (positive, negative, neutral). For Bangla, we utilized the Large Bangla Corpus spanning multiple domains. These datasets vary in domain and annotation quality, supporting comprehensive evaluation in low-resource settings.

4.2 Evaluation Metrics

We evaluated the metrics used for assessing Aspect-Based Sentiment Analysis (ABSA) models: Accuracy and F1-Score, selected to assess performance and address class imbalances. Accuracy measures the proportion of correct sentiment predictions (positive, negative, neutral) relative to all predictions, suitable for balanced datasets. The F1-Score, the harmonic mean of precision and recall, balances false positives and negatives, ideal for unbalanced datasets.

Model	Zero-shot		Few-shot		Fine-tuned	
	Acc	F1	Acc	F1	Acc	F1
BERT-Multilingual	0.244	0.321	0.774	0.777	0.795	0.792
RoBERTa-Twitter	0.718	0.733	0.803	0.802	0.816	0.815
XLM-RoBERTa	0.709	0.726	0.782	0.783	0.808	0.807
Bangla-BERT	0.295	0.303	0.427	0.287	0.509	0.441

Table 2: Performance on SemEval 2014 Laptop Corpus.

Model	Zero-shot		Few-shot		Fine-tuned	
	Acc	F1	Acc	F1	Acc	F1
BERT-Multilingual	0.170	0.224	0.783	0.786	0.778	0.778
RoBERTa-Twitter	0.686	0.696	0.796	0.798	0.791	0.790
XLM-RoBERTa	0.696	0.703	0.770	0.771	0.793	0.796
Bangla-BERT	0.571	0.419	0.576	0.421	0.576	0.421

Table 3: Performance on SemEval 2014 Restaurant Corpus.

4.3 Baseline Models

This subsection describes the baseline models evaluated for ABSA tasks, comprising transformer-based models, including one specifically pre-trained for Bangla. Each model is tested in zero-shot, few-shot, and fully fine-tuned settings to evaluate adaptability and performance across diverse datasets. The models include:

BERT-Multilingual: A multilingual BERT model pre-trained for sentiment analysis, suited for cross-lingual ABSA (nlptown/bert-base-multilingual-uncased-sentiment).

RoBERTa-Twitter: A RoBERTa model fine-tuned on Twitter data, effective for sentiment analysis in informal text `cardiffnlp/twitter-roberta-base-sentiment`.

XLM-RoBERTa: A cross-lingual RoBERTa model supporting multilingual sentiment analysis (cardiffnlp/twitter-xlm-roberta-base-sentiment).

Bangla-BERT: A BERT model pre-trained for Bangla, designed for low-resource language ABSA (sagorsarker/bangla-bert-base).

4.4 Evaluation Conditions

Models are evaluated in three settings to assess adaptability and performance: zero-shot, using pre-trained weights and prompts without task-specific training; few-shot, fine-tuned with approximately 10-100 labeled examples per sentiment class (positive, negative, neutral); and fully fine-tuned, trained on the entire labeled dataset for optimal supervised performance.

5 Results and Discussion

This section reports the performance of ABSA models in zero-shot, few-shot, and fully fine-tuned settings on SemEval 2014 Laptop and Restaurant datasets (English) and the Large Bangla ABSA Corpus. Metrics include accuracy and F1-score. The focus is to analyze the performance of the task-specific models and general-purpose LLMs in the ABSA task in contrast to the Gen-mABSA-T5, particularly in zero-shot settings.

5.1 Performance on SemEval 2014 Laptop Corpus

Table 2 summarizes the performance of baseline transformer models (BERT-Multilingual, RoBERTa-Twitter, XLM-RoBERTa, and Bangla-BERT) on the SemEval 2014 Laptop dataset in zero-shot, few-shot, and fine-tuned settings. In zero-shot, RoBERTa-Twitter achieves the highest accuracy (71.79 %) and F1 score (73.34 %), leveraging its pre-training on sentiment-rich Twitter data, while BERT-Multilingual lags at 24.36 % accuracy due to limited domain adaptation. Few-shot learning boosts performance across models, with RoBERTa-Twitter reaching 80.34 % accuracy, demonstrating the value of minimal labeled data for refinement. Fine-tuning yields further gains, peaking at 81.62 % accuracy for RoBERTa-Twitter, highlighting the effectiveness of full supervision on high-resource English data.

Our proposed framework achieves 73.50 % accuracy and 68.34 % F1 in zero-shot setting (Table 5), outperforming most baselines except

Model	Zero-shot		Few-shot		Fine-tuned	
	Acc	F1	Acc	F1	Acc	F1
BERT-Multilingual	0.237	0.347	0.945	0.951	0.947	0.943
RoBERTa-Twitter	0.539	0.688	0.926	0.922	0.927	0.927
XLM-RoBERTa	0.019	0.018	0.930	0.926	0.927	0.927
Bangla-BERT	0.615	0.641	0.615	0.662	0.863	0.850

Table 4: Performance on Large Bangla ABSA Corpus.

Model	Model Type	SemEval 2014 Laptop		SemEval 2014 Restaurant		Bangla Corpus	
		Acc	F1	Acc	F1	Acc	F1
BERT-Multilingual	Task-Specific	0.244	0.321	0.170	0.224	0.237	0.347
RoBERTa-Twitter	Task-Specific	0.718	0.733	0.686	0.696	0.539	0.688
XLM-RoBERTa	Task-Specific	0.709	0.726	0.696	0.703	0.002	0.002
Bangla-BERT	Task-Specific	0.295	0.303	0.571	0.419	0.615	0.614
gpt-oss-20b	General-Purpose	0.855	0.837	0.817	0.787	0.784	0.812
kimi-k2-instruct	General-Purpose	0.842	0.817	0.804	0.757	0.860	0.872
llama-3.1-8b-instant	General-Purpose	0.718	0.738	0.707	0.717	0.614	0.616
Gen-mABSA-T5	General-Purpose	0.735	0.683	0.736	0.668	0.616	0.628

Table 5: Zero-shot performance of general-purpose and task-specific models on ABSA.

RoBERTa Twitter and XLM-RoBERTa in task-specific pre-trained models. This underscores its generative zero-shot strength via prompt engineering and NLI framing without fine-tuning. Compared to general-purpose LLMs such as GPT-OSS-20B (20B parameters) (85.47 % accuracy with our designed prompt, Kimi-K2-Instruct (1T parameters) (84.19 %), and Llama-3.1-8B-Instant (8b parameters) (71.8%), our framework is commendable and lightweight.

These results demonstrate that prompt-engineered Flan-T5 can serve as a robust zero-shot baseline for English ABSA, approaching the performance of domain-specific pre-trained models without any task-specific adaptation. The gap to fine-tuned systems (approximately 5–8% F1) is expected and aligns with the zero-shot paradigm's goal of multilingual scalability rather than peak in-domain accuracy.

5.2 Performance on SemEval 2014 Restaurant Corpus

As shown in Table 3, baselines on the SemEval 2014 Restaurant dataset exhibits similar trends. XLM-RoBERTa leads in zero-shot (69.63 % accuracy, 70.28 % F1), benefiting from multilingual pre-training, while Bangla-BERT underperforms (57.07 %) due to language mismatch. Few-shot improvements are notable, with RoBERTa-Twitter at 79.58 % accuracy, and fine-tuning pushes XLM-RoBERTa to 79.32 %. These results indicate domain-specific challenges, such as varied aspect granularity in restaurant reviews.

Our framework attains 73.56 % accuracy and 66.81 % F1 in the zero-shot setting (Table 5), surpassing all task-specific models and Llama-3.1-8b-instant (70.7% accuracy) in general purpose model. Benchmarking against general-purpose LLMs, GPT-OSS-20B achieves 81.68 % accuracy and Kimi-K2-Instruct 80.37 % using our designed NLI-style concise and explicit prompt.

5.3 Performance on Large Bangla ABSA Corpus

On the Large Bangla Corpus, Gen-mABSA-T5 achieves the highest zero-shot accuracy (61.56%) across all task-specific models, and outperforms the Llama-3.1-8b-instant general-purpose model, despite using only 250M parameters and a tailored prompt. Though trailing larger general-purpose models like kimi-k2-instruct with 1 trillion parameters (Acc: 86%), Gen-mABSA-T5 remains highly reasonable, demonstrating that compact, prompt-engineered Flan-T5-base effectively handles Banglas morphological complexity in zero-shot ABSA, setting a new efficiency benchmark for low-resource languages.

5.4 Discussion and Observations

The experimental results reveal several key insights into the efficacy of zero-shot generative frameworks for multilingual ABSA, particularly in low-resource settings. First, consistently outperforms traditional task-specific transformer baselines in true zero-shot Bangla scenarios, demonstrating that prompt-engineered generative models can surpass language-specific discriminative models without any task-specific training. This advantage stems from Flan-T5's instruction-following capability and its exposure to NLI during pre-training, which allows ABSA to be naturally framed as a hypothesis–premise inference task.

Prompt ablation studies further confirm the critical role of prompt design. NLI-style prompts with concise and explicit instructions yield the highest average accuracy, outperforming direct prompts by 11–12 % and overly verbose prompts by 5–10 %. Adding domain context consistently degrades performance due to prompt dilution, reinforcing the principle that clarity and task focus are paramount in zero-shot settings.

Comparison with proprietary very large LLMs highlights a trade-off: GPT-OSS-20B (20B parameters) and Kimi-K2-Instruct (1T parameters), and Llama-8b-instant (8B parameters) achieve 61–85 % zero-shot accuracy on English SemEval tasks using conversational prompts, but require API access, higher latency, and non-deterministic outputs. Gen-mABSA-T5 (250M parameters), running locally on a single GPU, delivers 61-73 % accuracy with full reproducibility and no external dependencies, making it more suitable for resource-constrained research and deployment in low-resource regions.

6 Conclusion

Gen-mABSA-T5, a new multilingual zero-shot generative framework for Aspect-Based Sentiment Analysis (ABSA), demonstrates exceptional performance across English (SemEval 2014 Laptop and Restaurant) and Bangla (Large Bangla Corpus) datasets. It achieves a highest accuracy of 61.56% on the Large Bangla Corpus, improving to 86.32% with fine-tuning, while providing commendable accuracies of 73.50% (Laptop) and 73.56% (Restaurant) in English. Using prompt engineering and a generative transformer architecture, Gen-mABSA-T5 enables scalable, context-aware ABSA for low-resource languages like Bangla, addressing critical gaps in annotated resources and advancing multilingual sentiment analysis. Future work includes extending the framework to other underrepresented languages to further assess its generalizability

7 Limitations

The present study has several limitations that highlight areas for future improvement:

- The absence of a comprehensive and well-annotated Bangla ABSA data set limits the performance and generalization of the model due to data scarcity.
- The current prompts may not fully capture Bangla's complex morphological and syntactic characteristics, which can affect the precision of sentiment and aspect extraction.
- The model has not been evaluated on code-mixed or Romanized Bangla inputs (e.g., English and Bangla mixed sentences or Banglish), restricting its applicability in real-world multilingual communication contexts.

References

José I. Abreu Salas Agüero-Torales, Marvin M. and Antonio G. López-Herrera. 2021. Deep learning and multilingual sentiment analysis on social media data: An overview. *Applied Soft Computing 107 (2021): 107373.*

Md Farhan Ishmam Navid Hasin Alvee Md Shahnewaz Siddique Md Azam Hossain Alam, Sadia and Abu Raihan Mostofa Kamal. 2024. Bnsentmix: A diverse bengali-english code-mixed dataset for sentiment analysis. *arXiv preprint arXiv:2408.08964 (2024).*

Rajib Chandra Dey and Orvila Sarker. 2019. Sentiment analysis on bengali text using lexicon based approach. *22nd International Conference on Computer and Information Technology.*

Zichen He Yaqin Xie Huan, Hai and Zelin Guo. 2022. A multi-task dual-encoder framework for aspect sentiment triplet extraction. *IEEE Access 10 (2022): 103187-103199.*

Mohammad Barkatullah Junayed Hossain, Sheikh Md Abdullah and Md Fahad Monir. 2024. Exploring the efficacy of bert in bengali nlp: A study on sentiment analysis and aspect detection. *IEEE EUROCON 2023-20th International Conference on Smart Technologies, pp. 54-59. IEEE, 2023.*

Xingyu Fu Guangluan Xu Yang Yang Jiuniu Wang Li Jin Qing Liu Li, Xinlong and Tianyuan Xiang. 2020. Enhancing bert representation with context-aware embedding for aspect-based sentiment analysis. *IEEE Access 8 (2020): 46868-46876.*

Ayesha Tasnim Mahfuz Ahmed Masum, Sheikh Junayed Ahmed and Md Saiful Islam. 2020. Banabsa: An aspect-based sentiment analysis dataset for bengali and its baseline evaluation. *arXiv preprint arXiv:2012.00288 (2020).*

Shamsul Arafin Mahtab, Nazmul Islam, and Md Mahfuzur Rahaman. 2018. Sentiment analysis on bangladesh cricket with support vector machine. pages 1–4. IEEE, 2018 international conference on Bangla speech and language processing (ICBSLP).

Tanjim Mahmud Mohammad Shahadat Hossain Md Akash Rahman, Manoara Begum and Karl Andersson. 2024. Analyzing sentiments in elearning: A comparative study of bangla and romanized bangla text using transformers. *IEEE Access (2024).*

Dhiman Sarma Rishita Chakma Md Morshedul Islam, GM Sakhawat Hossain and Md Mynoddin. 2023. Deep-absa: A multichannel deep learning framework for aspect-based bangla sentiment analysis. *World Conference on Communication Computing (WCONF), pp. 1-6. IEEE, 2023.*

Azizkhan F. Pathan and Chetana Prakash. 2022. Attention-based position-aware framework for aspect-based opinion mining using bidirectional long short-term memory. *Journal of King Saud University-Computer and Information Sciences 34, no. 10 (2022): 8716-8726.*

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

Mumtahina Rahman Tuly Md Shihab Mahmud Rabeya, Mosa and Abdus Sattar. 2022. Sentiment analysis of bengali textual comments in field of sports using deep learning approach. *13th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-8. IEEE, 2022.*

Md Atikur Rahman and Emon Kumar Dey. Data 3, no. 2 (2018): 15. Datasets for aspect-based sentiment analysis in bangla and its baseline evaluation. *Applied Soft Computing 107 (2021): 107373.*

Md Atikur Rahman and Emon Kumar Dey. 2018. Datasets for aspect-based sentiment analysis in bangla and its baseline evaluation. *Data*, 3(2):15.

Shumaiya Akter Shammi, Sajal Das, Narayan Ranjan Chakraborty, Sumit Kumar Banshal, and Nishu Nath. 2023. A comprehensive roadmap on bangla text-based sentiment analysis. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–29.

Maksuda Haider Sayma Md. Mohsin Kabir Shihab Ahmed, Moythry Manir Samia and M. F. Mridha. 2024. trf-bert: A transformative approach to aspect-based sentiment analysis in the bengali language. *Plos one, 19(9), p.e0308050.*

Lei Shu, Hu Xu, Bing Liu, and Jiahua Chen. 2022. Zero-shot aspect-based sentiment analysis. *arXiv preprint arXiv:2202.01924*.

Yang Deng Xin Li Yifei Yuan Lidong Bing Zhang, Wenxuan and Wai Lam. 2021. Aspect sentiment quad prediction as paraphrase generation. *arXiv preprint arXiv:2110.00796 (2021).*

A Comprehensive Text Optimization Approach to Bangla Summarization

Irtifa Haider
Department of Computer
Science and Engineering
Jahangirnagar University,
Bangladesh
irtifa.stu2019@juniv.edu

Shanjida Alam
Department of Computer
Science and Engineering
Jahangirnagar University,
Bangladesh
shanjida.stu2019@juniv.edu

Md. Tazel Hossan
Department of Computer
Science and Engineering
Jahangirnagar University,
Bangladesh
tazel.stu2017@juniv.edu

Md. Musfique Anwar
Department of Computer
Science and Engineering
Jahangirnagar University,
Bangladesh
manwar@juniv.edu

Tanjim Taharat Aurpa
Department of Data
Science and Engineering
University of Frontier
Technology, Bangladesh
aurpa0001@uftb.ac.bd

Abstract

The task of Bengali text optimization demands not only the generation of concise and coherent summaries but also grammatical accuracy, semantic appropriateness, and factual reliability. This study presents a dual-phase optimization framework for Bengali text summarization that integrates entity-preserving preprocessing and abstractive generation with mT5, followed by refinement through sentence ranking, entity consistency enforcement, and optimization with instruction-tuned LLMs such as mBART. Evaluations using ROUGE, BLEU, BERTScore, and human ratings of fluency, adequacy, coherence, and readability show consistent gains over baseline summarizers. By embedding grammatical and factual safeguards into the summarization pipeline, this study establishes a robust and scalable benchmark for Bengali NLP, advancing text optimization research. Our model achieves 0.54 ROUGE-1 and 0.88 BERTScore on BANS-Data, outperforming recent multilingual baselines.

1 Introduction

In an era of information overload, producing clear, concise, and contextually relevant text is crucial for effective communication. *Text optimization*, refining summaries for readability, semantic accuracy, and purpose alignment, is increasingly essential. This includes enforcing grammatical soundness and precise meaning for diverse audiences in the digital age.

For Bangla, a morphologically rich language with a large speaker base yet limited NLP resources, optimization addresses unique challenges such as complex grammar (e.g., case markers, verb conjugations) to deliver high-quality, inclusive content. Despite its widespread use, Bangla lags behind high-resource languages in research and tooling, necessitating specialized models and pipelines (Haque et al., 2020).

This study tackles these issues by creating a two-phase pipeline that processes Bangla text, corrects errors (e.g., missing verbs, excessive conjunctions), and evaluates summaries with tailored metrics. Advances in Bangla NLP include large-scale resources such as a 100M-word corpus used for high-accuracy spell/grammar checking (Hossain et al., 2021) and XL-Sum's multilingual benchmark with Bangla coverage (Hasan et al., 2021).

Bangla's under-representation highlights the need for integrated optimization, unlike English's advanced LLM ecosystems, especially for domains such as news and healthcare. Early surveys such as Haque et al. (2020) catalog 14 Bangla summarization methods and emphasize resource gaps. Rahman et al. (2024) improve extractive ranking with Word2Vec embeddings but do not address grammar or factuality. Hasib et al. (2023) couple an extractive BenSumm stage with (Bangla)T5 for news, yet lack explicit faithfulness optimization. Miazee et al. (2025) explore neural abstractive pipelines but face scalability limits due to small datasets. On the grammar side, Hossain et al. (2024) (*Panini*) outperform BanglaT5 on a 7.7M-pair GEC corpus, while Sultana et al. (2024) show LLM potential for Bangla news summarization alongside cross-lingual/cultural gaps. Rule-based grammar pattern detection (Prapty et al., 2021) complements these efforts.

Therefore, we introduce a two-phase hybrid summarization–optimization model: **Phase 1** performs preprocessing, NER, BERT-style embed-

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 151–160
December 23, 2025 ©2025 Association for Computational Linguistics

dings, and uses mT5 to draft summaries; **Phase 2** optimizes with entity-aware ranking, redundancy reduction, and mBART-based refinement to improve readability, reduce hallucination, and preserve factual consistency.

We then combine automatic and human assessments: ROUGE (recall), BLEU (precision), and BERTScore (semantic similarity), plus human judgments on fluency, adequacy, coherence, and faithfulness. This research, based on the implemented pipeline, pursues the following objectives:

- Text Optimization via Grammar Correction (e.g., missing verbs, excessive conjunctions) and ensuring entity accuracy; assess with ROUGE, BLEU, and BERTScore.
- Two-Phase Hybrid Summarization Model combining classical NLP (tokenization, NER, stemming) with transformers (BERT embeddings, mT5 drafting, mBART refinement) to enhance readability and reduce redundancy.
- Comprehensive Evaluation using automated metrics (ROUGE, BLEU, BERTScore) and human evaluations (fluency, adequacy, coherence, faithfulness) to address prior Bangla ATS limitations.

By filling gaps in integrated text optimization, this work paves the way for future research in Bangla NLP, with potential to transform automated digital content for Bangla-speaking communities worldwide.

2 Related Work

The task of Automatic Text Summarization (ATS) for Bangla involves generating concise and coherent summaries from textual or structured data inputs. Research has evolved from traditional extractive methods toward optimization-driven and neural approaches, with growing interest in multi-document inputs (Haque et al., 2020; Wahab et al., 2024). Despite solid progress in Bangla NLP and optimization techniques, challenges persist due to limited datasets, linguistic complexity, and computational constraints (Haque et al., 2020; Hasan et al., 2021).

Several works have developed frameworks, surveys, and benchmarks tailored to Bangla, underscoring the importance of resource creation and evaluation (Haque et al., 2020; Hossain et al., 2021; Hasan et al., 2023). While progress is visible, Bangla remains under-represented in large-scale multilingual corpora and lacks mature optimization-driven pipelines—clear priorities for future work (Hasan et al., 2021; Wahab et al., 2024).

Early Bangla efforts emphasized extractive techniques for their simplicity and modest resource demand. Haque et al. (2020) survey 14 Bangla summarization methods and outline evaluation hurdles and resource scarcity. Rahman et al. (2024) compare frequency/rule-based/Word2Vec extractive models and show embeddings better capture semantics, though their scope remains ranking-only without grammar or factual checks. Foysal et al. (2021) introduce Bangla-ExtraSum, comparing five extractive approaches (incl. transformer-based and Word2Vec-assisted models) on 200 prior + 500 new articles with dual human references; they report strong scores (e.g., F1 ≈ 0.68/0.63 on two sets) but stay strictly extractive. Khan et al. (2023) present a hybrid extractive model combining NER, keywords, POS, and sentence cues over a curated 960-passage, 8-domain dataset (2,880 human summaries), achieving competitive precision/F1 yet still within extractive, traditional metrics.

To address multi-document needs, Hasan et al. (2023) release BUSUM-BNLP, a 1,000-article update-summarization corpus across six domains with human references; simple TF-IDF baselines outperform SentenceRank there, but the work remains extractive and ranking-focused.

With neural architectures, Bangla abstraction gains traction. Miazee et al. (2025) propose an abstractive pipeline with neural modeling and preprocessing; data scale and evaluation remain modest. Hayat et al. (2023) benchmark multiple transformer variants (incl. fine-tuned BanglaT5) and report best ROUGE-2 ≈ 13.83 for BanglaT5. Hasib et al. (2023) build a hybrid pipeline where an extractive BenSumm stage feeds (Bangla)T5; the hybrid beats direct T5 on news, though evaluation is mostly ROUGE/BLEU. Beyond news, Barsha and Uddin (2023) compare BanglaT5 vs. pointer-generator networks for Tagore short-story summarization; feature-augmented pointer-generator wins on ROUGE, highlighting the value of linguistically informed architectures for literary text.

Foundational resources for correctness run in parallel. Hossain et al. (2021) develop large-scale Bangla spell/grammar checking (100M-word corpus; 1M-word lexicon) with strong accuracy but outside summarization. Prapty et al. (2021) apply CFG+CYK for Bangla grammar pattern detection, effective yet rule-dependent. Hossain et al. (2024) introduce *Panini*, a transformer-based GEC trained on a 7.7M parallel corpus, outperforming BanglaT5; while not a summarizer, such GEC can be integrated into summarization post-editing for fluency and correctness.

Optimization methods underpin ATS from heuristic selection to meta-heuristics. Wahab et al. (2024) review optimization-driven ATS (e.g., GA, PSO), stressing accuracy–cost trade-offs and real-time constraints. Classical probabilistic approaches (e.g., Naïve Bayes + topic words) illustrate optimization flavors in single-syllable languages (Thu, 2014), though mostly handcrafted and extractive. At the training level, optimizer choice (Adam, RMSProp, etc.) materially affects ROUGE and convergence for abstractive news summarization (Kumari et al., 2023).

Broader multilingual/cross-domain work informs Bangla directions. Hasan et al. (2021) release XL-Sum (44 languages incl. Bangla), a key benchmark, though Bangla remains relatively small/noisy. Semantic generalization with deep models improves rare-word handling in English settings (Kouris et al., 2019). Domain-specific Bangla evaluations with LLMs emerge in health (Abrar et al., 2024) and question generation (Faieaz et al., 2025), while multimodal chart-to-text for Bangla is newly explored in Chart-Summ (Tanjila et al., 2025). For Bangla news, comparative studies of LM/LLMs indicate potential but also gaps in faithfulness and robustness (Sultana et al., 2024). Overall, recent LLM-based summarization highlights controllability and scale, yet Bangla still needs optimization-guided, resource-aware adaptation and evaluation beyond ROUGE (e.g., factuality, hallucination, and grammar checks) (Wahab et al., 2024; Hasib et al., 2023; Hayat et al., 2023). Recent multilingual models such as BanglaT5, Qwen2.5-Instruct, and LLaMA-3-Instruct demonstrate promising zero-shot summarization capabilities.

3 Methodology

This study proposes a two-phase framework for Bengali text summarization: summarization and optimization (Figure 1). The pipeline combines classical NLP with transformer-based and instruction-tuned LLMs. Preprocessing removes noise (stopwords, URLs, digits), while a Named Entity Recognition (NER) module preserves key entities.

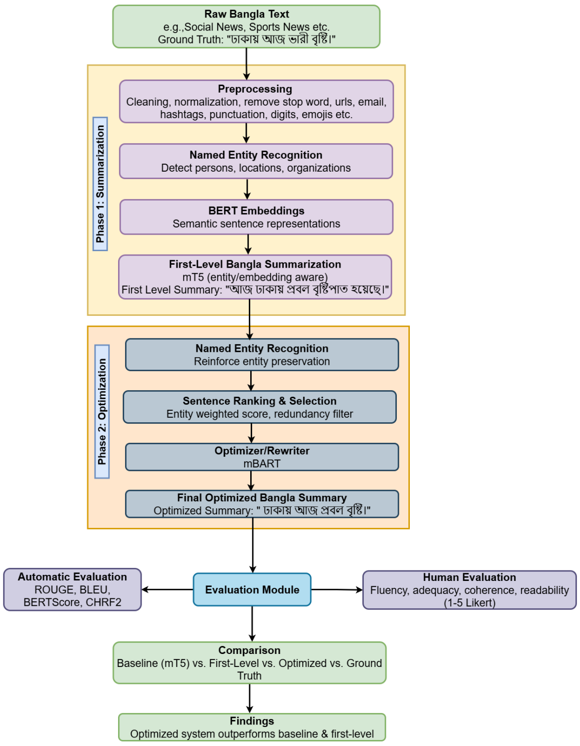

Raw Bangla Text
e.g.,Social News, Sports News etc.
Ground Truth: "ঢাকায় আজ ভারী বৃষ্টি।"
Phase 1: Summarization
Preprocessing
Cleaning, normalization, remove stop word, urls, email, hashtags, punctuation, digits, emojis etc.
Named Entity Recognition
Detect persons, locations, organizations
BERT Embeddings
Semantic sentence representations
First-Level Bangla Summarization
mT5 (entity/embedding aware)
First Level Summary: "আজ ঢাকায় প্রবল বৃষ্টিপাত হয়েছে।"
Phase 2: Optimization
Named Entity Recognition
Reinforce entity preservation
Sentence Ranking & Selection
Entity weighted score, redundancy filter
Optimizer/Rewriter
mBART
Final Optimized Bangla Summary
Optimized Summary: " ঢাকায় আজ প্রবল বৃষ্টি।"
Automatic Evaluation
ROUGE, BLEU, BERTScore, CHRF2
Evaluation Module
Human Evaluation
Fluency, adequacy, coherence, readability (1-5 Likert)
Comparison
Baseline (mT5) vs. First-Level vs. Optimized vs. Ground Truth
Findings
Optimized system outperforms baseline & first-level

Figure 1: Text Optimization Framework for Bengali Text

3.1 Summarization Techniques

Bengali summarization methods fall into two main categories: extractive, which select important sentences from the source, and abstractive, which generate new sentences that condense meaning. Early work relied on rule-based heuristics (e.g., sentence position, keywords) or machine learning models using features such as TF–IDF and entity frequency. While these methods are efficient, they often miss deeper semantics. Abstractive approaches, on the other hand, leverage neural architectures. Initial RNN+attention models struggled with long sequences, but transformer-based models like mT5 and mBART have since improved fluency and semantic coverage, producing more natural summaries.

3.2 Optimization with LLMs

Recent advances incorporate instruction-tuned large language models (LLMs) to refine draft summaries into coherent, concise, and entity-faithful outputs. In our work, we employ mBART for intermediate optimization. There are also larger models such as LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct for refining draft summaries while preserving entities and handling complex structures, removing redundancy, and producing concise outputs under strict length constraints.

3.3 Phase - 1 : Summarization

3.3.1 Dataset and Preprocessing

We use the Bengali Abstractive News Summarization Dataset (BANSData) (Bhattacharjee et al., 2021), containing 19,096 articles and corresponding human-written summaries across domains such as politics, economy, sports, and culture. Articles range from 5–76 words, with summaries of 3–12 words, making the dataset well-suited for abstractive summarization. A preprocessing pipeline was applied to clean and normalize the corpus by removing URLs, digits, hashtags, and redundant punctuation. Tokenization was performed using BNLP and NLTK tokenizers, stopwords were removed via the BengaliCorpus list, and BanglaStemmer was used for morphological normalization. To ensure factual accuracy, a Named Entity Recognition (NER) module preserved critical entities (persons, locations, organizations) throughout the pipeline, yielding an entity-aware dataset for embedding, summarization, and optimization.

3.3.2 Named Entity Recognition (NER)

NER is central to the framework for preserving factual reliability in low-resource Bengali summarization. Using BNLP's BengaliNER toolkit, entities such as “বাংলাদেশ ব্যাংক” (Bangladesh Bank) and “ঢাকা” (Dhaka) are detected and explicitly protected from stemming or removal. During optimization, an additional verification step ensures that all entities present in the source are retained in the final summary. This integration minimizes semantic drift and enhances the trustworthiness of generated outputs.

3.3.3 Sentence Embeddings

To capture semantic depth, each preprocessed sentence is encoded into dense vector representations using BERT embeddings, which model both left and right contexts. These embeddings serve three purposes: (i) reduce redundancy by detecting semantically overlapping sentences, (ii) strengthen ranking by promoting diverse content coverage, and (iii) enrich abstractive summarization with contextualized features that complement entity preservation. Combined with NER, this dual-layered approach ensures factual reliability, coherence, and a strong foundation for high-quality Bengali summaries.

3.3.4 Summarization with mT5

The first stage employs the mT5 model, a multilingual sequence-to-sequence transformer pre-trained on diverse languages, including Bengali. Leveraging embeddings and preserved entities, mT5 generates draft summaries that are fluent and contextually meaningful. For instance, it can condense “জাপানের প্রধানমন্ত্রী শিনজো আবের সরকার নতুন করে একটি বাজেট অনুমোদন করেছে… (“Japan’s Prime Minister Shinzo Abe’s government has approved a new budget…”) into shorter, coherent forms. However, drafts may still contain redundancy, factual drift, or incomplete coverage, requiring refinement in Phase 2.

3.4 Phase - 2 : Optimization

3.4.1 Sentence Ranking and Selection

To refine mT5 drafts, a ranking module filters sentences based on heuristic criteria. Sentences containing named entities are prioritized through an entity weight, while a length score penalizes those that are overly short or verbose. Redundancy is reduced using BERT embeddings to remove semantic overlaps, and semantic diversity is encouraged to ensure broad coverage of different aspects. Together, these steps yield summaries that are concise, entity-aware, and semantically rich, forming the basis for final optimization.

3.4.2 Entity Preservation Enforcement

Before final rewriting, BNLP's BengaliNER toolkit verifies entity consistency (e.g., “বাংলাদেশ ব্যাংক” (Bangladesh Bank), “ঢাকা” (Dhaka)). Missing or altered entities are corrected, preventing factual distortion and ensuring reliable summaries.

3.5 Optimizer/Rewriter (mBART)

The last stage uses mBART, a multilingual sequence-to-sequence transformer pre-trained with denoising objectives. Given the source article and entity-preserved draft, it:

- Enhances readability with fluent Bengali,
- Removes residual redundancy,
- Preserves named entities, and
- Improves semantic alignment with the source.

3.6 Evaluation Metrics

Evaluation of system-generated summaries is performed using both automatic metrics and human judgment. To ensure robust assessment, we adopt widely recognized metrics: ROUGE, BLEU, and BERTScore. These metrics collectively capture lexical overlap, n-gram precision, semantic similarity, and character-level fluency, providing a comprehensive evaluation of both raw summaries and optimized summaries.

3.6.1 Automatic Evaluation Metrics

To ensure robust assessment, we employ these complementary evaluation metrics:

- **ROUGE-N** measures recall by evaluating n-gram overlaps (e.g., unigrams, bigrams) between system and reference summaries.
- **BLEU** measures precision of n-gram matches, originally for machine translation, with a brevity penalty to avoid overly short outputs.
- **BERTScore** computes semantic similarity using contextual embeddings (e.g., BERT, BanglaBERT) and cosine similarity between tokens.

Final optimized summaries, generated by the LLM refinement stage (mBART), were assessed using these four automatic metrics.

3.7 Human Evaluation

Three native Bengali linguists (each with $\geq$ 3 years of experience in NLP or linguistics) independently rated a stratified random sample of 200 summaries (balanced across domains) on a 5-point Likert scale for fluency, adequacy, coherence, and readability. The raters were blinded to system identity, and ties were allowed. Inter-annotator agreement was measured using Krippendorff's α (ordinal) for each dimension; disagreements were resolved by majority vote. We report per-dimension means ± 95% CIs and α. Where:

- Fluency: grammatical correctness and sentence naturalness.
- Adequacy: extent to which the draft captured essential content.
- Coherence: logical progression and organization.
- Readability: overall ease of comprehension.

The evaluation strategy was designed to assess the effectiveness of both phases of the proposed framework. Specifically, human evaluation was applied after Phase 1 (Summarization) to assess draft summaries generated by mT5 and refined by sentence ranking, while automatic metrics were applied after Phase 2 (Optimization) to benchmark the final optimized summaries against human-written references.

4 Experimental Setup and Evaluation

The framework was meticulously tested on the Bengali Abstractive News Summarization Dataset (BANSData), a robust corpus comprising 19,096 article-summary pairs, utilizing models such as mT5 and mBART. The dataset was initially split into four parts, which underwent iterative refinement. Performance was systematically assessed using a suite of automatic metrics—ROUGE-1, ROUGE-2, ROUGE-L, BLEU, and BERTScore—alongside detailed human evaluations focusing on fluency, adequacy, coherence, and readability, scored on a 1–5 Likert scale by a panel of three expert native Bangla linguists with linguistic experience.

4.1 Experimental Setup

Dataset. We employ the Bengali Abstractive News Summarization Dataset (BANSData), a curated collection of Bengali news articles and their corresponding human-written summaries obtained from `bangla.bdnews24.com`. The corpus comprises 19,096 article–summary pairs across diverse domains such as politics, economy, sports, culture, and social issues, and has been widely used in prior Bangla summarization research. Key hyperparameters for both phases are summarized in Table 1. Settings follow standard Hugging Face Transformer defaults, with mixed-precision training on a single NVIDIA A100 GPU.

Baseline Model. The baseline system is based on the sequence-to-sequence Long Short-Term Memory (LSTM) network with attention proposed by (Bhattacharjee et al., 2021). The model employs an encoder–decoder architecture with local attention to generate fluent and human-like abstractive summaries from Bangla news articles. The authors also released the Bengali Abstractive News Summarization (BANS) dataset, the largest publicly available Bangla summarization corpus, comprising over 19,000

Parameter	**mT5 (Phase 1)**	**mBART (Phase 2)**
Optimizer	AdamW (β_1=0.9, β_2=0.999, ϵ=1 × 10^{-8})	AdamW (same)
Learning rate	3×10^{-4}	1×10^{-5}
Warm-up / Decay	Linear (5%) / Linear	Linear (same)
Weight decay	0.01	0.01
Label smoothing	0.1	0.1
Batch size	8 × 8 (accum.=64)	8 × 8 (accum.=64)
Epochs (max)	5 (early stop patience 3)	2
Steps (approx.)	∼35k	∼10k
Token limits	512 (input), 100 (output)	512 (input), 100 (output)
GPU	A100 (40 GB)	A100 (40 GB)
Seed	42	42
Dataset split	80 / 10 / 10	80 / 10 / 10

Table 1: Fine-tuning hyperparameters for mT5 (Phase 1) and mBART (Phase 2). Both configurations use linear warm-up (5%), linear decay, and fixed seed = 42. Experiments were conducted on BANSData (train/dev/test = 80/10/10) using a single A100 GPU.

news articles and human-written summaries collected from bangla.bdnews24.com and published on Kaggle. Their system demonstrated competitive performance in both quantitative metrics (BLEU, ROUGE) and human evaluation, establishing a strong benchmark for BANSData. For comparison, this work adopts csebuetnlp/mT5_multilingual_XLSum, a multilingual T5 variant fine-tuned for abstractive summarization. The model is configured with a maximum input length of 512 tokens, maximum output length of 100 tokens, a minimum output of 30 tokens, a six-beam search, and a no-repeat 3-gram constraint to reduce redundancy. Summaries are generated using a standardized pipeline comprising tokenization, abstractive generation, and post-processing, with outputs stored in CSV format for subsequent evaluation.

Evaluation. Performance was assessed using ROUGE-1, ROUGE-L, BLEU, and BERTScore. In addition, three native Bengali linguists provided human ratings on fluency, adequacy, coherence, and readability using a five-point Likert scale. This combination of automatic and human evaluation established a robust baseline for the subsequent optimization experiments.

4.2 Automated Metric Results

Figure 2 presents the quantitative evaluation results in terms of ROUGE-1, ROUGE-L, BLEU, and BERTScore across the baseline, first-level, and optimized summaries. The baseline system yielded modest scores (ROUGE-1: 0.30, ROUGE-L: 0.31, BLEU: 0.30, BERTScore: 0.52). First-level summaries demonstrated consistent improvements, with notable gains in BLEU (0.42) and BERTScore (0.55), alongside moderate increases in ROUGE metrics. The optimized summaries achieved the best performance overall, reaching ROUGE-1: 0.54, ROUGE-L: 0.36, BLEU: 0.50, and BERTScore: 0.88. These findings indicate that progressive optimization not only improves lexical overlap but also substantially enhances semantic fidelity, with BERTScore showing the largest relative gain. Recent Bangla and multilingual summarization baselines were also evaluated for comparison, as summarized in Table 2.

Model (Configuration)	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**BLEU**	**BERTScore (F1)**
BanglaT5 (fine-tuned; Hayat et al. 2023)	0.41	0.14	0.39	0.32	0.82
Qwen2.5-7B-Instruct (0-shot)	0.44	0.16	0.40	0.35	0.84
LLaMA-3-Instruct (0-shot)	0.43	0.15	0.39	0.34	0.83
mT5 (Phase 1; ours)	**0.46**	**0.17**	**0.33**	**0.42**	**0.55**
mT5 → mBART (Phase 2; ours)	**0.54**	**0.20**	**0.36**	**0.50**	**0.88**

Table 2: Comparison on BANSData between the proposed dual-phase optimization framework and recent Bangla/multilingual baselines.

As shown in Table 2, the proposed framework delivers substantial gains over both Bangla-specific and multilingual baselines. Compared with BanglaT5, Phase 2 yields +0.13 ROUGE-1 and +0.18 BLEU improvements, while surpassing instruction-tuned LLaMA-3 and Qwen 2.5 models across all metrics. These results highlight the effectiveness of the two-phase optimization strategy in achieving superior lexical precision, semantic consistency, and factual reliability within Bengali abstractive summarization.

4.3 Human Evaluation Results

Figure 3 reports average human ratings (1 to 5 Likert scale) across four dimensions: fluency, ade-

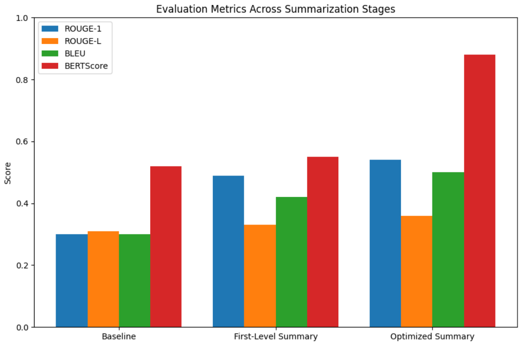


Figure 2: Automated metric results

quacy, coherence, and readability. The first-level summaries received mid-scale ratings between 3.8 to 4.1 on average, indicating reasonable but imperfect quality. Optimized summaries consistently outperformed them, scoring 4.3 to 4.5 across all dimensions. Gains were most notable in fluency and readability, suggesting that the optimization stage effectively improved grammatical correctness, sentence naturalness, and ease of comprehension. Adequacy and coherence also showed consistent improvements, highlighting the benefit of entity preservation and redundancy reduction in the final optimization phase.

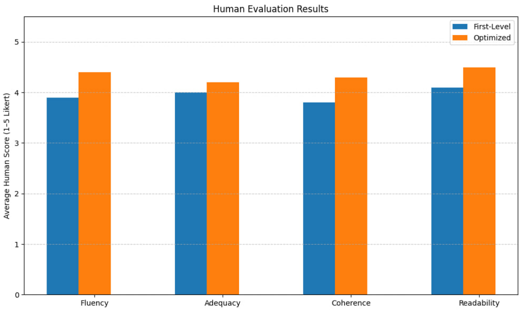


Figure 3: Human evaluation scores across fluency, adequacy, coherence, and readability

The Figure 4 (radar chart) visualizes the ratings for fluency, coherence, content preservation, and relevance across different models. The proposed model outperforms all baseline models, especially in content preservation and relevance, which are critical for generating useful and informative summaries.

Taken together, human and automatic evaluations demonstrate that the proposed two-phase framework yields consistent improvements over both the baseline and first-level drafts. Human judges valued the readability and fluency gains, while automatic scores highlighted advances in semantic alignment and lexical coverage. This dual evidence supports the effectiveness of combining entity-aware draft generation with LLM-based optimization for Bangla summarization.

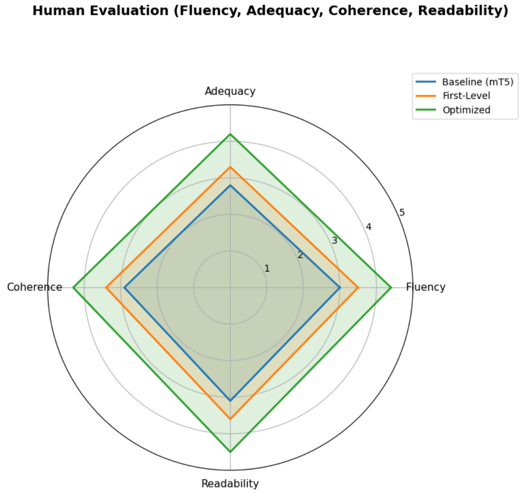


Figure 4: Radar chart of human evaluation scores (1–5) for Fluency, Adequacy, Coherence, and Readability across Baseline (mT5), First-Level, and Optimized models

4.4 Text Sample Comparison

To illustrate the qualitative impact of the proposed framework, sample outputs from the three stages of summarization are compared against the human-written reference summary. The examples highlight how the system evolved from the baseline through the first-level summarizer to the final optimized summary. To better understand the behavior of each stage, a stratified sample of 50 summaries from the baseline, Phase 1 (mT5), and the final optimized system was manually examined. Three recurring error types were observed: (i) entity omissions or substitutions, where key actors (e.g., organizations or locations) are missing or replaced by generic phrases; (ii) hallucinated details, where numbers, dates, or causal explanations not supported by the source article are introduced; and (iii) over-compression, where summaries become too short and drop essential background information. The second-phase optimization, which combines entity-aware ranking with mBART refinement, substantially reduces the first two categories (especially for named entities such as banks and cities) and slightly increases summary length, thereby mitigating over-compression while preserving the main event described in the article.

The experimental findings clearly demonstrate

Type	Text
Source Text	“বাংলাদেশ ব্যাংক ঘোষণা করেছে যে নতুন অর্থবছরের কৃষি খাতে ঋণের পরিমাণ বাড়ানো হবে। এর ফলে কৃষকরা স্বল্পসুদে ঋণ নিতে পারবেন।” (Bangladesh Bank has announced that the amount of loans in the agricultural sector will be increased in the new fiscal year. As a result, farmers will be able to take loans at low interest rates.)
Reference Summary (Human)	“কৃষি খাতে ঋণ বাড়াবে বাংলাদেশ ব্যাংক।” (Bangladesh Bank will increase loans in the agricultural sector.)
First-Level Summary	“বাংলাদেশ ব্যাংক নতুন অর্থবছরে কৃষি খাতে ঋণ বাড়াবে।” (Bangladesh Bank will increase agricultural loans in the new fiscal year.)
Optimized Summary	“বাংলাদেশ ব্যাংক কৃষকদের জন্য কৃষি খাতে ঋণ বাড়ানোর ঘোষণা দিয়েছে।” (Bangladesh Bank has announced an increase in agricultural loans for farmers.)

Table 3: Example of text sample comparison across phases

the effectiveness of the proposed two-phase optimization framework. Starting from the baseline model, which produced modest results, the incorporation of entity-aware embeddings and heuristic ranking in the first-level summarizer delivered measurable improvements in informativeness and coherence. The final optimization stage, leveraging NER reinforcement, redundancy filtering, and mBART rewriting, achieved the strongest performance across both automated metrics and human evaluation, substantially surpassing the baseline and intermediate systems. Automated results confirm notable gains in ROUGE, BLEU, and BERTScore, while human assessments highlight significant improvements in fluency, adequacy, coherence, and readability. These outcomes validate the central claim of this study: that integrating classical preprocessing, semantic embeddings, and instruction-tuned LLM refinement produces Bangla summaries that are not only concise and fluent but also factually reliable and stylistically aligned with human-written references.

5 Conclusion

5.1 Our Findings

The framework was evaluated on the Bengali Abstractive News Summarization Dataset (BANSData). Performance was assessed using ROUGE-1, ROUGE-L, BLEU, and BERTScore, complemented by human evaluation from three expert native Bangla linguists who rated fluency, adequacy, coherence, and readability on a five-point Likert scale.

The baseline model achieved ROUGE-1 0.30, ROUGE-L 0.31, BLEU 0.30, and BERTScore 0.52, with mean human ratings around 3.8. Incorporating entity-aware embeddings and a ranking mechanism in the first-level summarizer improved all metrics, especially BLEU (from 0.30 to 0.42) and BERTScore (from 0.52 to 0.55); human ratings increased to about 4.0. The Phase 2 optimized system, enhanced through NER-based entity preservation and mBART-driven semantic refinement, achieved ROUGE-1 0.54, ROUGE-L 0.36, BLEU 0.50, and BERTScore 0.88. Mean human evaluation scores exceeded 4.4 across all dimensions, confirming the framework’s effectiveness in improving both lexical overlap and semantic fidelity.

Taken together, the automatic metrics and human evaluations confirm that the proposed two-phase framework substantially outperforms both the baseline and intermediate systems. Improvements averaged around +0.20 for ROUGE-1 and ROUGE-L, +0.20 for BLEU, and +0.36 for BERTScore. Human evaluation scores increased by approximately 0.6 points on the five-point Likert scale, reflecting consistent gains in fluency, adequacy, coherence, and readability. These findings validate the effectiveness of combining entity-aware draft generation with LLM-based optimization for producing high-quality Bangla abstractive summaries that achieve stronger lexical overlap and semantic fidelity.

5.2 Future Prospects

Further fine-tuning of mBART on domain-specific Bangla corpora could help mitigate trade-offs between brevity and semantic richness. Incorporating real-time data sources such as `bangla.bdnews24.com` may enhance adaptability to evolving news content. Expanding collaboration with a broader panel of linguists could also refine evaluation criteria and improve the balance between fluency and adequacy.

Limitations. The proposed framework is trained and evaluated on BANSData news articles, which may limit generalization to conversational or literary Bangla without domain adaptation. Nevertheless, the two-phase architecture itself is model- and domain-agnostic and can be extended to other Bangla genres (e.g., healthcare, education, social media) given appropriate training data and domain adaptation. Although entity preservation mitigates factual drift, no external fact-checking module is applied, and errors in Named Entity Recognition could propagate. Human evaluation is conducted with a small group of expert linguists using Likert-scale ratings, which, despite reporting inter-annotator agreement, remains inherently subjective. Furthermore, optimization with large language models increases computational cost, and efficiency-oriented techniques such as quantization or distillation are not yet explored.

References

Ajwad Abrar, Farzana Tabassum, and Sabbir Ahmed. 2024. Performance evaluation of large language models in bangla consumer health query summarization. In *Proceedings of the 2024 27th International Conference on Computer and Information Technology (ICCIT)*, pages 2748–2753, Dhaka, Bangladesh. IEEE.

Fahmida Afroja Hoque Barsha and Mohammed Nazim Uddin. 2023. Comparative analysis of BanglaT5 and pointer generator network for bengali abstractive story summarization. In *Proceedings of the 2023 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, pages 84–88, Dhaka, Bangladesh. IEEE.

Prithwiraj Bhattacharjee, Avi Mallick, Md. Saiful Islam, and Marium-E-Jannat. 2021. Bengali abstractive news summarization (bans): A neural attention approach. In *Proceedings of International Conference on Trends in Computational and Cognitive Engineering*, pages 41–51, Singapore. Springer Singapore.

Washik Wali Faieaz, Sayma Jannat, Pronoy Kumar Mondal, Shahriar Shadman Khan, Shuvo Karmaker, and Md. Sadekur Rahman. 2025. Advancing bangla NLP: Transformer-based question generation using fine-tuned LLM. In *Proceedings of the 2025 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–7, Chittagong, Bangladesh. IEEE.

Towhid Ahmed Foysal, Mohaimen Abid Mahadi, Md. Mahadi Hasan Nahid, and Ayesha Tasnim. 2021. Bangla-extrasum: Comparative analysis of different methods in automated extractive bengali text summarization. In *Proceedings of the 2021 5th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pages 1–6, Dhaka, Bangladesh. IEEE.

Md. Majharul Haque, Suraiya Pervin, Anowar Hossain, and Zerina Begum. 2020. Approaches and trends of automatic bangla text summarization: Challenges and opportunities. *International Journal of Technology Diffusion (IJTD)*, 11(4):17–29.

Md. Nahid Hasan, Rafsan Bari Shafin, Marwa Khanom Nurtaj, Zeshan Ahmed, M. Saddam Hossain Khan, Rashedul Amin Tuhin, and Md. Mohsin Uddin. 2023. Implementation of bangla extractive update summarization task on BUSUM-BNLP dataset: A multi-document update summarization corpus. In *Proceedings of the 2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–6, Istanbul, Turkiye. IEEE.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Samin Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. Xl-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Khan Md Hasib, Md. Atiqur Rahman, Mustavi Ibne Masum, Friso De Boer, Sami Azam, and Asif Karim. 2023. Bengali news abstractive summarization: T5 transformer and hybrid approach. In *Proceedings of the 2023 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 539–545, Port Macquarie, NSW, Australia. IEEE.

S. M. Afif Ibne Hayat, Avishek Das, and Mohammed Moshiul Hoque. 2023. Abstractive bengali text summarization using transformer-based learning. In *Proceedings of the 2023 6th International Conference on Electrical Information and Communication Technology (EICT)*, pages 1–6, Khulna, Bangladesh. IEEE.

Nahid Hossain, Mehedi Hasan Bijoy, Salekul Islam, and Swakkhar Shatabda. 2024. Panini: A transformer-based grammatical error correction method for bangla. *Neural Computing and Applications*, 36:3463–3477.

Nahid Hossain, Salekul Islam, and Mohammad Nurul Huda. 2021. Development of bangla spell and grammar checkers: Resource creation and evaluation. *IEEE Access*, 9:141079–141097.

Alam Khan, Sanjida Akter Ishita, Fariha Zaman, Ashiqul Islam Ashik, and Md. Moinul Hoque. 2023. Intelligent combination of approaches towards improved bangla text summarization. In *Proceedings of the 2023 International Conference on Applied Intelligence and Sustainable Computing (ICAISC)*, pages 1–6, Rajshahi, Bangladesh. IEEE.

Panagiotis Kouris, Georgios Alexandridis, and Andreas Stafylopatis. 2019. Abstractive text summarization based on deep learning and semantic content generalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5082–5092, Florence, Italy. Association for Computational Linguistics.

Namrata Kumari, Nikhil Sharma, and Pradeep Singh. 2023. Performance of optimizers in text summarization for news articles. In *Procedia Computer Science*, volume 218, pages 2430–2437. Elsevier.

Asif Ahammad Miazee, Tonmoy Roy, Md Robiul Islam, and Yeamin Safat. 2025. Abstractive text summarization for bangla language using NLP and machine learning approaches. *arXiv preprint arXiv:2501.15051*.

Aroni Saha Prapty, Md. Rifat Anwar, and K. M. Azharul Hasan. 2021. A rule-based parsing for bangla grammar pattern detection. In *Proceedings of International Joint Conference on Advances in Computational Intelligence*, pages 319–331, Singapore. Springer Singapore.

Mizanur Rahman, Sajib Debnath, Masud Rana, Saydul Akbar Murad, Abu Jafar Md. Muzahid, Syed Zahidur Rashid, and Abdul Gafur. 2024. Bangla text summarization analysis using machine learning: An extractive approach. In *Proceedings of the 2nd Human Engineering Symposium (HUMENS 2023)*, pages 65–80, Singapore. Springer.

Faria Sultana, Md. Tahmid Hasan Fuad, Rahat Rizvi Rahman, Md. Hossain, Md. Ashraful Amin, Asif Rahman, and Amin Ahsan. 2024. How good are LM and LLMs in bangla newspaper article summarization? In *Pattern Recognition. ICPR 2024. Lecture Notes in Computer Science*, volume 15320, pages 77–91, Cham. Springer Nature.

Nahida Akter Tanjila, Afrin Sultana Poushi, Sazid Abdullah Farhan, Abu Raihan Mostofa Kamal, Md. Azam Hossain, and Md. Hamjajul Ashmafee. 2025. Bengali chartsumm: A benchmark dataset and study on feasibility of large language models on bengali chart to text summarization. In *Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL)*, pages 35–45, Singapore. Association for Computational Linguistics.

Ha Nguyen Thi Thu. 2014. An optimization text summarization method based on naïve bayes and topic word for single syllable language. *Applied Mathematical Sciences*, 8(3):99–115.

Muhammad Hafizul H. Wahab, Nor Hafiza Ali, Nor Asilah Wati Abdul Hamid, Shamala K. Subramaniam, Rohaya Latip, and Mohamed Othman. 2024. A review on optimization-based automatic text summarization approach. *IEEE Access*, 12:4892–4909.

BLUCK: A Benchmark Dataset for Bengali Linguistic Understanding and Cultural Knowledge

Daeen Kabir[*†], **Minhajur Rahman Chowdhury Mahim**[*◇], **Sheikh Shafayat**[◇], **Adnan Sadik**[◇], **Arian Ahmed**[◇], **Eunsu Kim**[◇], **Alice Oh**[◇‡]

[†]Independent Researcher, [◇]KAIST

daeenkabirdk03@gmail.com
{minhaj, sheikh.shafayat, adnansadik235, arian.ahmed, kes0317}@kaist.ac.kr
alice.oh@kaist.edu

Abstract

In this work, we introduce **BLUCK**, a new dataset designed to measure the performance of Large Language Models (LLMs) in Bengali linguistic understanding and cultural knowledge. Our dataset comprises 2366 multiple-choice questions (MCQs) carefully curated from compiled collections of several college and job level examinations and spans 23 categories covering knowledge on Bangladesh's culture and history and Bengali linguistics. We benchmarked BLUCK using 6 proprietary and 3 open-source LLMs - including GPT-4o, Claude-3.5-Sonnet, Gemini-1.5-Pro, Llama-3.3-70B-Instruct, and DeepSeekV3. Our results show that while these models perform reasonably well overall, they, however, struggles in some areas of Bengali phonetics. Although current LLMs' performance on Bengali cultural and linguistic contexts is still not comparable to that of mainstream languages like English, our results indicate Bengali's status as a mid-resource language. Importantly, BLUCK is also the first MCQ-based evaluation benchmark that is centered around native Bengali culture, history, and linguistics.

Categories		No. of Questions
History	Ancient Bengal	99
	British Bengal	40
	Pakistan Era	106
Culture	Indigenous People	31
	Arts, Heritage & Media	69
	National Issues	15
	Constitution	31
	Resources	36
	Geography	87
	Law	284
Phonetics	Alphabet	10
	Pronunciation	69
	Conjunct Letters	23
	Sound & Letters	48
	Sound Changes	54
	Phonetic Combining Rules	184
	Miscellaneous Phonetics	80
Semantics	Synonyms	364
	Antonyms	165
	One Word Expressions	180
	Idioms	198
	Proverbs	47
	Miscellaneous	146
Total		2366

Table 1: Statistics of BLUCK

1 Introduction

Recently, Large Language Models (LLMs) have demonstrated remarkable success in multilingual capabilities. In the case of Bengali, OpenAI's O1 model achieved an impressive score of 0.873 (OpenAI et al., 2024b) on the MMLU benchmark. However, most evaluations of Bengali, including MMLU, rely on translated English datasets assessing general knowledge skills or focus exclusively on STEM fields, such as math and science (Shafayat et al., 2024). Despite the growing emphasis on evaluations that capture cultural and linguistic contexts for LLMs, the performance of models in Bengali-specific cultural knowledge or reasoning skills remains unexplored.

Given that Bengali is the 7th most spoken language in the world, with over 237 million native speakers, it is crucial to address the lack of high-quality Bengali-specific evaluation datasets. To this end, we introduce **BLUCK**[1], a Benchmark Dataset for Bengali Linguistic Understanding and Cultural Knowledge. Through a rigorous curation process—encompassing careful annotation, multiple rounds of cross-inspection, and digitization—we have compiled a dataset of 2,366 multiple-choice questions (MCQs) that encompass

[*] Equal contribution.

[†] Daeen Kabir conducted this research during his undergraduate studies at KAIST.

[‡] Corresponding author

[1] Dataset link: BLUCK

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 161–179
December 23, 2025 ©2025 Association for Computational Linguistics

extensive knowledge of the culture, history, and language of Bangladesh, into 23 subcategories. Table 1 presents the overall statistics and categories of BLUCK.

Our evaluation of 9 LLMs using BLUCK offers valuable insights into the current status of LLMs in understanding Bengali language and cultural knowledge. Specifically, GPT-4o and Claude-3.5-Sonnet achieve the highest scores, around 73% in a 0-shot setting—approximately 7% lower than their performance on the MMLU benchmark. Overall, the models tend to perform well in the history category but show weaker results in the culture category, particularly on national issues. Similarly, in the phonetics category, their performance is generally low, with GPT-4o scores of 0.377 in pronunciation and 0.407 in sound changes. The lower performance in specific categories, such as culture and phonetics, highlights the current models' limitations in Bengali-specific knowledge. These findings underscore the potential for improvement in these areas, providing valuable insights for the future development of Bengali language models.

2 Related Works

2.1 Cultural Sensitive Dataset

There has been a growing effort to create culturally sensitive benchmarks for evaluating LLMs across different languages and regions. Datasets like CULTURALBENCH (Chiu et al., 2024), CLIcK (Kim et al., 2024) for Korean cultural knowledge RoCulturaBench (Masala et al., 2024) for Romanian culture are designed to assess LLMs' ability to understand cultural context beyond linguistic fluency. Similarly, BLEnD (Myung et al., 2024) provides a multilingual cultural dataset from more than 13 different languages.

Despite these advances, Bengali remains significantly underrepresented in cultural-sensitive datasets. Most cultural evaluation benchmarks focus on high-resource languages or specific regional cultures, leaving a major gap in Bengali cultural and linguistic understanding.

2.2 Bengali Dataset

Several benchmarks have been developed to evaluate LLMs in general and multilingual tasks, but only a few focus on specific Bengali knowledge. M-MMLU (OpenAI, 2024) and Global-MMLU (Singh et al., 2024) are some of the few well-known benchmarks that include Bengali in their multilingual evaluation settings. Nevertheless, their Bengali questions are mostly translated from English, limiting their effectiveness in assessing native-level linguistic understanding.

For Bengali reasoning tasks, MGSM (Shi et al., 2022) provided the Bengali translations of GSM8K (Cobbe et al., 2021), one of the prominent datasets for grade school math problems along with other languages, although its scope remains limited. Later, BEnQA (Shafayat et al., 2024) provided multiple choice questions as official English-Bengali corpus, sourced from Bangladesh's national board exams, focusing on STEM subjects. More recently, Tiger LLM (Raihan and Zampieri, 2025) aggregated diverse Bengali corpora, including educational, literary, and QA datasets, contributing significantly to Bengali NLP resources. Notably, there is still no phonetics-focused dataset for Bengali, which leaves granular but crucial aspects of the language absent from current evaluations.

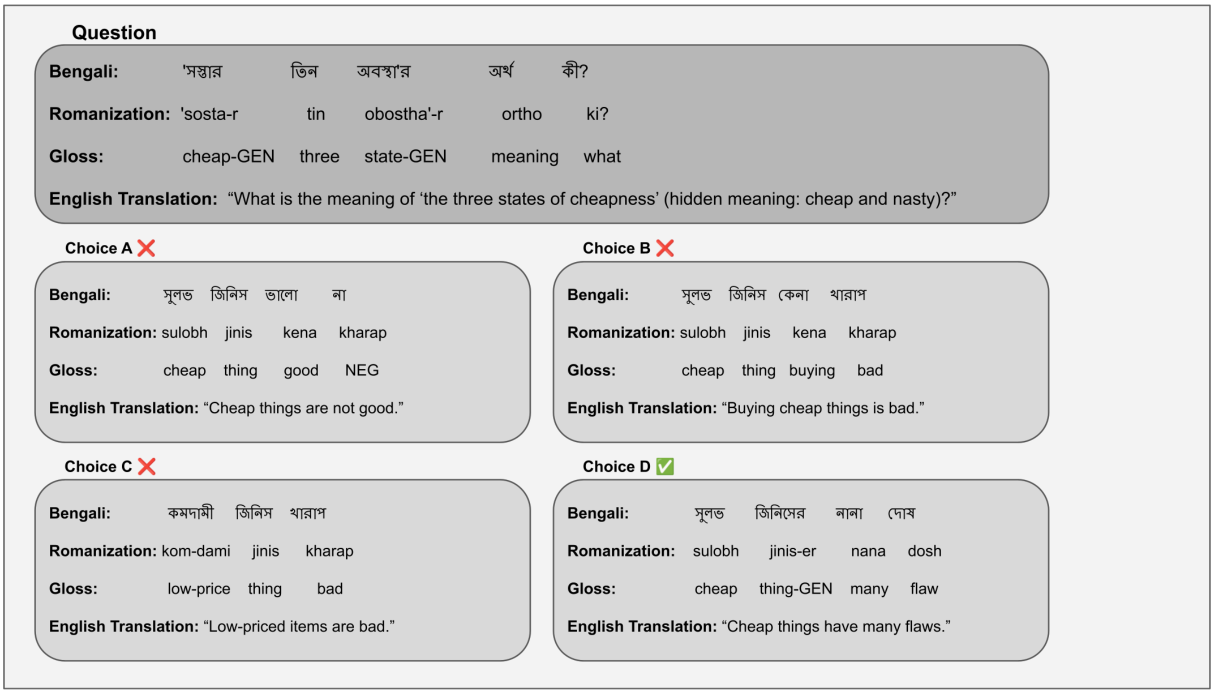
To the best of our knowledge, BLUCK is the first comprehensive benchmark to include Bengali-related reasoning and knowledge questions, focusing on Bengali history, culture, and language. BLUCK is specifically designed to evaluate LLMs in native Bengali contexts, complementing existing multilingual and subject-specific benchmarks.

3 BLUCK: A Benchmark Dataset for Bengali Linguistic Understanding and Cultural Knowledge

BLUCK contains immersive knowledge organized into these four major domains: Bangladesh's history, Bangladeshi culture, Bengali phonetics, and Bengali semantics. A summary of these categories, along with the corresponding number of questions is provided in Table 1.

3.1 Data Collection

Data is collected from publicly available printed copies of previous examination papers from the following sources: a) Bangladesh Civil Service (BCS) Examinations, b) university entrance examinations in Bangladesh, c) Bangladesh Bar Council Preliminary Examinations, d) bank job examinations, and e) several public job examinations. These official examinations are selected for their reliability and authoritative assessment of general knowledge in Bangladesh. These exams consist of extensive native knowledge on Bangladesh's history, culture, law, language, and various other



Question

Bengali: 'সস্তার তিন অবস্থা'র অর্থ কী?

Romanization: 'sosta-r tin obostha'-r ortho ki?

Gloss: cheap-GEN three state-GEN meaning what

English Translation: “What is the meaning of ‘the three states of cheapness’ (hidden meaning: cheap and nasty)?”

Choice A ❌

Bengali: সুলভ জিনিস ভালো না

Romanization: sulobh jinis kena kharap

Gloss: cheap thing good NEG

English Translation: “Cheap things are not good.”

Choice B ❌

Bengali: সুলভ জিনিস কেনা খারাপ

Romanization: sulobh jinis kena kharap

Gloss: cheap thing buying bad

English Translation: “Buying cheap things is bad.”

Choice C ❌

Bengali: কমদামী জিনিস খারাপ

Romanization: kom-dami jinis kharap

Gloss: low-price thing bad

English Translation: “Low-priced items are bad.”

Choice D ✅

Bengali: সুলভ জিনিসের নানা দোষ

Romanization: sulobh jinis-er nana dosh

Gloss: cheap thing-GEN many flaw

English Translation: “Cheap things have many flaws.”

Figure 1: Illustration of a MCQA sample from the Semantics domain. To aid linguistic understanding of non-native Bengali speakers, we include the original Bengali script, romanization, and word-by-word glossing, following the Leipzig Glossing Rules, alongside the English translation.

academic disciplines. For BLUCK's creation, we select only the MCQs and follow a question selection criterion, based on which we omit these types of questions: a) fact-based questions loosely representing Bangladesh's history, culture, and language b) questions on contemporary issues in Bangladesh (to ensure long-term relevance), c) insignificant date-related or ‘number-based answer option’ questions (to avoid arbitrary or trivial answers).

3.2 Dataset Curation

(1) Categorization After data collection, we categorize by utilizing general knowledge and Bengali language guidebooks that organize questions similar to the ones in our dataset. This approach ensures proper categorization for some of the categories in the culture domain, and allows us to group similar categories into the four main domains of our dataset.

(2) Two Round Inspection The preliminary question selection task is distributed among the authors by category. To ensure quality, we implement two rounds of double-blind cross-checking. Independently, a second author collects questions for the same categories. The two authors then cross-check and agree on a preliminary set. In the second round, two more authors inspect these agreed-upon preliminary set in a double-blind manner, and cross-check their selection lists. All authors ensure that their selection process conforms to the aforementioned selection criteria . This process ensures that our dataset contains high-quality questions representing the history and culture of Bangladesh, and its rich linguistic knowledge.

(3) Digitization After inspection, professional annotators, proficient in Bengali, digitize the MCQs for easier access and manipulation of the data. This is done to minimize the errors when digitized. To finalize our dataset, we conduct refinement: a) cleaning duplicate and inconsistent entries, b) correcting existing typing errors, and c) final checking to remove erroneous questions. This extensive approach ensures reliability and proper representation of the categories in our dataset.

4 Experiment

4.1 Experimental Setup

In order to evaluate LLMs' performance on the history and culture of Bangladesh, and Bengali phonetics and semantics, we conduct experiments on the BLUCK dataset using both proprietary and open-source models. We utilized the following LLMs:

- **Proprietary models**: GPT-4o, GPT-4o-mini (OpenAI et al., 2024a) [2], Claude-3.5-Sonnet, Claude-3.5-Haiku [3], Gemini-1.5-Pro, Gemini-1.5-Flash (Team et al., 2024) [4]
- **Open-source models**: Llama-3.1-8B-Instruct, Llama-3.3-70B-Instruct (Grattafiori et al., 2024), DeepSeekV3 (DeepSeek-AI et al., 2024)

Since BLUCK consists largely of factual-knowledge based questions, we conduct evaluation without any chain-of-thought (CoT) reasoning, using both zero-shot and five-shot settings. As shown in Figure 3a in Appendix 7, for prompt, we utilize system and user prompts, explicitly instructing the model to output only the option 'letter' in order to save API computational cost (Petrov et al., 2023). Following the criteria in KoBBQ (Jin et al., 2024), we only accept generated responses based on: a) response with only the alphabet as answer, b) response mentioning term corresponding to one of the options, iii) response convey the answer in the form 'answer:', or 'answer is', etc. Responses showing signs of hallucination, or producing bizarre outputs such as single Bengali letter as response are omitted.

4.2 Result

Our evaluation results are summarized in Table 2, which highlights the performance scores for all 23 categories of our dataset for the major models. Table 4 in Appendix 7 shows the same for the small-sized language models. Our results indicate that Claude-3.5-Sonnet, GPT-4o, Gemini-1.5-pro, and DeepSeekV3 demonstrate considerable knowledge of Bangladeshi history and the semantics of Bengali language. However, all the models struggle with phonetics, especially in areas such as pronunciation and sound changes. Claude-3.5-Sonnet emerges as the best overall model with consistent performance across all categories in both settings. It's performance in Bengali phonetics, which is the most difficult category, is 10% better than the 2nd best model in this domain. GPT-4o closely follows, performing the best in history, culture, and semantics, while Claude-3.5-Sonnet achieves best performance in culture and phonetics. The smaller models exhibit surprisingly reasonable performance, with Gemini-1.5-Flash and Claude-3.5-Haiku surpassing even Llama-3.3-70B-Instruct in 5-shot setting. Llama-3.1-8B-Instruct, on the other hand, lags behind all other smaller models, showing very limited performance overall.

5 Discussions

The benchmark results reveal significant variations in model performance across different categories and shot settings. Firstly, it is visible that 5-shot prompting leads to notable performance improvements (between 5% to 10%) across all models, which aligns with the findings that large language models pick up categorical cues from the examples and reduce the 'search space' for MCQ solution under few-shot settings (Brown et al., 2020).

Secondly, proprietary models like GPT-4o and Claude-3.5-Sonnet consistently outperform open-source models for most of the categories, suggesting that the former have a stronger contextual understanding of Bengali.

In addition, 'Pronunciation', and 'Sound Changes' are notable categories in which models exhibit poor performance. This strongly suggests that phonetic nuances in Bengali still remain underrepresented in existing LLMs, even with few-shot prompting.

The findings, overall, reinforce the need for more robust culture sensitive Bengali resources in LLM pretraining and evaluation benchmarks to improve performance in underrepresented Bengali linguistic and cultural areas.

5.1 CoT Evaluation on Pronunciation

We identify existing LLMs as performing markedly poor in the pronunciation category. To understand the source of this weakness, we conducted an additional experiment prompting Claude-3.5-Sonnet to use CoT reasoning on this particular subset under a zero-shot setting. Table 3 shows the results.

The CoT approach did not yield considerable improvement. After evaluating the CoT responses, we attribute this mainly to insufficient knowledge of Bengali grammar as well as limited understanding of Bengali phonetic rules. In several cases, the model made an error in the very first reasoning step due to incorrect background assumptions, causing the entire reasoning chain to collapse. We also observed a tendency for the model to fixate on an

[2] We use GPT-4o-2024-08-06 and GPT-4o-mini-2024-07-18 version using OPENAI API.

[3] We use Claude-3.5-Sonnet-20241022 and Claude-3.5-Haiku-20241022 version using Anthropic API.

[4] We use gemini-1.5-pro-Latest, gemini-1.5-flash-latest using Gemini API key in Google AI Studio.

Categories		GPT-4o		Claude-3.5-Sonnet		Gemini-1.5-Pro		Llama-3.3-70B		DeepSeekV3	
		0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot
History	Ancient Bengal	0.899	**0.919**	0.879	0.889	0.758	0.758	0.687	0.677	0.859	0.889
	British Bengal	0.925	**0.975**	0.875	0.9	0.675	0.95	0.8	0.85	0.9	0.95
	Pakistan Era	0.745	**0.783**	0.67	0.764	0.481	0.613	0.5	0.509	0.717	0.726
	Average	0.837	**0.869**	0.788	0.837	0.624	0.727	0.624	0.633	0.804	0.829
Culture	Indigenous People	0.806	0.839	0.871	**0.935**	0.516	0.71	0.516	0.742	0.774	0.871
	Arts, Heritage & Media	0.739	**0.768**	0.725	0.696	0.58	0.594	0.58	0.551	0.725	**0.768**
	National Issues	0.467	0.467	0.733	**0.8**	0.6	0.733	0.2	0.6	0.467	0.6
	Constitution	0.806	0.871	0.871	0.935	0.806	0.903	0.677	0.71	0.935	**0.968**
	Resources	0.778	0.778	0.722	0.778	0.5	0.639	0.528	0.583	0.694	**0.806**
	Geography	0.828	**0.862**	0.759	0.793	0.598	0.724	0.655	0.621	0.701	0.77
	Law	0.68	0.715	0.648	**0.718**	0.613	0.715	0.496	0.588	0.588	0.641
	Average	0.725	**0.758**	0.707	**0.758**	0.604	0.707	0.537	0.604	0.656	0.718
Phonetics	Alphabet	0.6	0.7	0.6	**0.9**	0.2	**0.9**	0.6	**0.9**	0.6	**0.9**
	Pronunciation	0.377	0.406	0.348	**0.507**	0.246	0.391	0.217	0.333	0.29	0.348
	Conjunct Letters	0.652	0.739	0.826	**0.957**	0.652	0.826	0.739	0.826	0.696	0.826
	Sound & Letters	0.771	0.729	0.708	**0.792**	0.625	0.771	0.542	0.688	0.688	0.75
	Sound Changes	0.407	0.611	0.5	**0.667**	0.463	0.63	0.352	0.537	0.407	0.574
	Phonetic Combining Rules	0.516	0.603	0.663	**0.761**	0.533	0.609	0.446	0.473	0.609	0.63
	Miscellaneous Phonetics	0.638	0.675	0.588	**0.7**	0.5	0.588	0.463	0.575	0.575	0.675
	Average	0.538	0.609	0.596	**0.718**	0.485	0.609	0.432	0.526	0.545	0.618
Semantics	Synonyms	0.874	0.912	0.893	**0.923**	0.769	0.835	0.676	0.772	0.852	0.907
	Antonyms	0.782	**0.891**	0.855	0.879	0.733	0.812	0.685	0.739	0.77	0.848
	One Word Expressions	0.717	0.811	0.778	0.806	0.589	0.661	0.556	0.6	0.717	**0.828**
	Idioms	0.722	**0.808**	0.652	0.747	0.606	0.662	0.495	0.505	0.626	0.697
	Proverbs	0.787	0.83	0.83	**0.894**	0.723	0.809	0.638	0.745	0.766	0.787
	Miscellaneous	**0.733**	0.712	0.692	0.719	0.575	0.589	0.486	0.514	0.678	0.719
	Average	0.785	**0.844**	0.795	0.837	0.677	0.738	0.598	0.655	0.750	0.817
Overall Average		0.727	0.780	0.735	**0.795**	0.617	0.704	0.554	0.615	0.693	0.756

Table 2: BLUCK benchmark comparison by subcategories and major categories across major models in 0-shot and 5-shot settings. The highest accuracy(s) for each category are boldy marked.

answer prematurely and then generate a forced rationale to justify it. Cases are listed in the Appendix 7.

Pronunciation	Claude 3.5-Sonnet		
	0-shot	5-shot	**0-shot CoT**
Accuracy	0.348	0.507	0.478

Table 3: Claude 3.5 performance on the Pronunciation subcategory.

6 Conclusion

We introduced BLUCK, a linguistic and culture-sensitive Bengali dataset, locally sourcing from official college and job-level examinations in Bangladesh. BLUCK provides a diverse set of 2366 multiple-choice questions that fall under 23 subcategories organized across four domains. Our evaluation using state-of-the-art LLMs showcases their knowledge in historical and semantics aspects of Bengali, while exposes their weakness in linguistically nuanced areas. Future research should expand BLUCK and improve LLMs' understanding of Bengali linguistic and cultural nuances.

Limitations

We acknowledge certain limitations in our work. Since our dataset consists solely of text-based questions, we cannot determine whether the models arrived at their answers through reasoning processes different from those of humans. Moreover, given the richness of Bengali culture, history, and linguistic diversity, as well as the growing importance of M-MMLU, Global-MMLU and other large-scale multilingual benchmarks, our contribution remains relatively small in comparison. However, we hope that BLUCK serves as a stepping stone to improve Bengali culture-sensitive LLM research.

Ethical Considerations

The BLUCK dataset is fully available and has been manually curated and reviewed to mitigate any chance of having harmful contents. This dataset will be publicly accessible and distributed under the CC BY-SA 4.0 license. Our work has been reviewed and received approval from the Institutional Review Board (IRB) at our institution. All annotators involved in this project were compensated above the minimum wage and standards. Finally, AI-assisted tools were used solely for grammar and language refinement. They were not used for writing, analysis, or coding in any capacity.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Yu Ying Chiu, Liwei Jiang, Bill Yuchen Lin, Chan Young Park, Shuyue Stella Li, Sahithya Ravi, Mehar Bhatia, Maria Antoniak, Yulia Tsvetkov, Vered Shwartz, and Yejin Choi. 2024. Culturalbench: a robust, diverse and challenging benchmark on measuring the (lack of) cultural knowledge of llms. *Preprint*, arXiv:2410.02677.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins,

Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wen-

wen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jiho Jin, Jiseon Kim, Nayeon Lee, Haneul Yoo, Alice Oh, and Hwaran Lee. 2024. Kobbq: Korean bias benchmark for question answering. *Preprint*, arXiv:2307.16778.

Eunsu Kim, Juyoung Suk, Philhoon Oh, Haneul Yoo, James Thorne, and Alice Oh. 2024. CLIcK: A benchmark dataset of cultural and linguistic intelligence in Korean. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3335–3346, Torino, Italia. ELRA and ICCL.

Mihai Masala, Denis C. Ilie-Ablachim, Alexandru Dima, Dragos Corlatescu, Miruna Zavelca, Ovio Olaru, Simina Terian, Andrei Terian, Marius Leordeanu, Horia Velicu, Marius Popescu, Mihai Dascalu, and Traian Rebedea. 2024. "vorbeşti româneşte?" a recipe to train powerful romanian llms with english instructions. *Preprint*, arXiv:2406.18266.

Junho Myung, Nayeon Lee, Yi Zhou, Jiho Jin, Rifki Putri, Dimosthenis Antypas, Hsuvas Borkakoty, Eunsu Kim, Carla Perez-Almendros, Abinew Ali Ayele, Victor Gutierrez Basulto, Yazmin Ibanez-Garcia, Hwaran Lee, Shamsuddeen H Muhammad, Kiwoong Park, Anar Rzayev, Nina White, Seid Muhie Yimam, Mohammad Taher Pilehvar, Nedjma Ousidhoum, Jose Camacho-Collados, and Alice Oh. 2024. Blend: A benchmark for llms on everyday knowledge in diverse cultures and languages. In *Advances in Neural Information Processing Systems*, volume 37, pages 78104–78146. Curran Associates, Inc.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024a. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024b. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2024. Multilingual massive multitask language understanding (mmmlu).

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems*.

Nishat Raihan and Marcos Zampieri. 2025. TigerLLM - a family of Bangla large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 887–896, Vienna, Austria. Association for Computational Linguistics.

Matthew Renze and Erhan Guven. 2024. The effect of sampling temperature on problem solving in large language models. *Preprint*, arXiv:2402.05201.

Sheikh Shafayat, H Hasan, Minhajur Mahim, Rifki Putri, James Thorne, and Alice Oh. 2024. BEnQA: A question answering benchmark for Bengali and English. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1158–1177, Bangkok, Thailand. Association for Computational Linguistics.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Shivalika Singh, Angelika Romanou, Clémentine Fourrier, David I. Adelani, Jian Gang Ngui, Daniel Vila-Suero, Peerat Limkonchotiwat, Kelly Marchisio, Wei Qi Leong, Yosephine Susanto, Raymond Ng, Shayne Longpre, Wei-Yin Ko, Madeline Smith, Antoine Bosselut, Alice Oh, Andre F. T. Martins, Leshem Choshen, Daphne Ippolito, Enzo Ferrante, Marzieh Fadaee, Beyza Ermis, and Sara Hooker. 2024. Global mmlu: Understanding and addressing cultural and linguistic biases in multilingual evaluation. *Preprint*, arXiv:2412.03304.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, Andrea Tacchetti, Colin Gaffney, Samira Daruki, Olcan Sercinoglu, Zach Gleicher, Juliette Love, Paul Voigtlaender, Rohan Jain, Gabriela Surita, Kareem Mohamed, Rory Blevins, Junwhan Ahn, Tao Zhu, Kornraphop Kawintiranon, Orhan Firat, Yiming Gu, Yujing Zhang, Matthew Rahtz, Manaal Faruqui, Natalie Clay, Justin Gilmer, JD Co-Reyes, Ivo Penchev, Rui Zhu, Nobuyuki Morioka, Kevin Hui, Krishna Haridasan, Victor Campos, Mahdis Mahdieh, Mandy Guo, Samer Hassan, Kevin Kilgour, Arpi Vezer, Heng-Tze Cheng, Raoul de Liedekerke, Siddharth Goyal, Paul Barham, DJ Strouse, Seb Noury, Jonas Adler, Mukund Sundararajan, Sharad Vikram, Dmitry Lepikhin, Michela Paganini, Xavier Garcia, Fan Yang, Dasha Valter, Maja Trebacz, Kiran Vodrahalli, Chulayuth Asawaroengchai, Roman Ring, Norbert Kalb, Livio Baldini Soares, Siddhartha Brahma, David Steiner, Tianhe Yu, Fabian Mentzer, Antoine He, Lucas Gonzalez, Bibo Xu, Raphael Lopez Kaufman, Laurent El Shafey, Junhyuk Oh, Tom Hennigan, George van den Driessche, Seth Odoom, Mario Lucic, Becca Roelofs, Sid Lall, Amit Marathe, Betty Chan, Santiago Ontanon, Luheng He, Denis Teplyashin, Jonathan Lai, Phil Crone, Bogdan Damoc, Lewis Ho, Sebastian Riedel, Karel Lenc, Chih-Kuan Yeh, Aakanksha Chowdhery, Yang Xu, Mehran Kazemi, Ehsan Amid, Anastasia Petrushkina, Kevin Swersky, Ali Khodaei, Gowoon Chen, Chris Larkin, Mario Pinto, Geng Yan, Adria Puigdomenech Badia, Piyush Patil, Steven Hansen, Dave Orr, Sebastien M. R. Arnold, Jordan Grimstad, Andrew Dai, Sholto Douglas, Rishika Sinha, Vikas Yadav, Xi Chen, Elena Gribovskaya, Jacob Austin, Jeffrey Zhao, Kaushal Patel, Paul Komarek, Sophia Austin, Sebastian Borgeaud, Linda Friso, Abhimanyu Goyal, Ben Caine, Kris Cao, Da-Woon Chung, Matthew Lamm, Gabe Barth-Maron, Thais Kagohara, Kate Olszewska, Mia Chen, Kaushik Shivakumar, Rishabh Agarwal, Harshal Godhia, Ravi Rajwar, Javier Snaider, Xerxes Dotiwalla, Yuan Liu, Aditya Barua, Victor Ungureanu, Yuan Zhang, Bat-Orgil Batsaikhan, Mateo Wirth, James Qin, Ivo Danihelka, Tulsee Doshi, Martin Chadwick, Jilin Chen, Sanil Jain, Quoc Le, Arjun Kar, Madhu Gurumurthy, Cheng Li, Ruoxin Sang, Fangyu Liu, Lampros Lamprou, Rich Munoz, Nathan Lintz, Harsh Mehta, Heidi Howard, Malcolm Reynolds, Lora Aroyo, Quan Wang, Lorenzo Blanco, Albin Cassirer, Jordan Griffith, Dipanjan Das, Stephan Lee, Jakub Sygnowski, Zach Fisher, James Besley, Richard Powell, Zafarali Ahmed, Dominik Paulus, David Reitter, Zalan Borsos, Rishabh Joshi, Aedan Pope, Steven Hand, Vittorio Selo, Vihan Jain, Nikhil Sethi, Megha Goel, Takaki Makino, Rhys May, Zhen Yang, Johan Schalkwyk, Christina Butterfield, Anja Hauth, Alex Goldin, Will Hawkins, Evan Senter, Sergey Brin, Oliver Woodman, Marvin Ritter, Eric Noland, Minh Giang, Vijay Bolina, Lisa Lee, Tim Blyth, Ian Mackinnon, Machel Reid, Obaid Sarvana, David Silver, Alexander Chen, Lily Wang, Loren Maggiore, Oscar Chang, Nithya Attaluri, Gregory Thornton, Chung-Cheng Chiu, Oskar Bunyan, Nir Levine, Timothy Chung, Evgenii Eltyshev, Xiance Si, Timothy Lillicrap, Demetra Brady, Vaibhav Aggarwal, Boxi Wu, Yuanzhong Xu, Ross McIlroy, Kartikeya Badola, Paramjit Sandhu, Erica Moreira, Wojciech Stokowiec, Ross Hemsley, Dong Li, Alex Tudor, Pranav Shyam, Elahe Rahimtoroghi, Salem Haykal, Pablo Sprechmann, Xiang Zhou, Diana Mincu, Yujia Li, Ravi Addanki, Kalpesh Krishna, Xiao Wu, Alexandre Frechette, Matan Eyal, Allan Dafoe, Dave Lacey, Jay Whang, Thi Avrahami, Ye Zhang, Emanuel Taropa, Hanzhao Lin, Daniel Toyama, Eliza Rutherford, Motoki Sano, HyunJeong Choe, Alex Tomala, Chalence Safranek-Shrader, Nora Kassner, Mantas Pajarskas, Matt Harvey, Sean Sechrist, Meire Fortunato, Christina Lyu, Gamaleldin Elsayed, Chenkai Kuang, James Lottes, Eric Chu, Chao Jia, Chih-Wei Chen, Peter Humphreys, Kate Baumli, Connie Tao, Rajkumar Samuel, Cicero Nogueira dos Santos, Anders Andreassen, Nemanja Rakićević, Dominik Grewe,

Aviral Kumar, Stephanie Winkler, Jonathan Caton, Andrew Brock, Sid Dalmia, Hannah Sheahan, Iain Barr, Yingjie Miao, Paul Natsev, Jacob Devlin, Feryal Behbahani, Flavien Prost, Yanhua Sun, Artiom Myaskovsky, Thanumalayan Sankaranarayana Pillai, Dan Hurt, Angeliki Lazaridou, Xi Xiong, Ce Zheng, Fabio Pardo, Xiaowei Li, Dan Horgan, Joe Stanton, Moran Ambar, Fei Xia, Alejandro Lince, Mingqiu Wang, Basil Mustafa, Albert Webson, Hyo Lee, Rohan Anil, Martin Wicke, Timothy Dozat, Abhishek Sinha, Enrique Piqueras, Elahe Dabir, Shyam Upadhyay, Anudhyan Boral, Lisa Anne Hendricks, Corey Fry, Josip Djolonga, Yi Su, Jake Walker, Jane Labanowski, Ronny Huang, Vedant Misra, Jeremy Chen, RJ Skerry-Ryan, Avi Singh, Shruti Rijhwani, Dian Yu, Alex Castro-Ros, Beer Changpinyo, Romina Datta, Sumit Bagri, Arnar Mar Hrafnkelsson, Marcello Maggioni, Daniel Zheng, Yury Sulsky, Shaobo Hou, Tom Le Paine, Antoine Yang, Jason Riesa, Dominika Rogozinska, Dror Marcus, Dalia El Badawy, Qiao Zhang, Luyu Wang, Helen Miller, Jeremy Greer, Lars Lowe Sjos, Azade Nova, Heiga Zen, Rahma Chaabouni, Mihaela Rosca, Jiepu Jiang, Charlie Chen, Ruibo Liu, Tara Sainath, Maxim Krikun, Alex Polozov, Jean-Baptiste Lespiau, Josh Newlan, Zeyncep Cankara, Soo Kwak, Yunhan Xu, Phil Chen, Andy Coenen, Clemens Meyer, Katerina Tsihlas, Ada Ma, Juraj Gottweis, Jinwei Xing, Chenjie Gu, Jin Miao, Christian Frank, Zeynep Cankara, Sanjay Ganapathy, Ishita Dasgupta, Steph Hughes-Fitt, Heng Chen, David Reid, Keran Rong, Hongmin Fan, Joost van Amersfoort, Vincent Zhuang, Aaron Cohen, Shixiang Shane Gu, Anhad Mohananey, Anastasija Ilic, Taylor Tobin, John Wieting, Anna Bortsova, Phoebe Thacker, Emma Wang, Emily Caveness, Justin Chiu, Eren Sezener, Alex Kaskasoli, Steven Baker, Katie Millican, Mohamed Elhawaty, Kostas Aisopos, Carl Lebsack, Nathan Byrd, Hanjun Dai, Wenhao Jia, Matthew Wiethoff, Elnaz Davoodi, Albert Weston, Lakshman Yagati, Arun Ahuja, Isabel Gao, Golan Pundak, Susan Zhang, Michael Azzam, Khe Chai Sim, Sergi Caelles, James Keeling, Abhanshu Sharma, Andy Swing, YaGuang Li, Chenxi Liu, Carrie Grimes Bostock, Yamini Bansal, Zachary Nado, Ankesh Anand, Josh Lipschultz, Abhijit Karmarkar, Lev Proleev, Abe Ittycheriah, Soheil Hassas Yeganeh, George Polovets, Aleksandra Faust, Jiao Sun, Alban Rrustemi, Pen Li, Rakesh Shivanna, Jeremiah Liu, Chris Welty, Federico Lebron, Anirudh Baddepudi, Sebastian Krause, Emilio Parisotto, Radu Soricut, Zheng Xu, Dawn Bloxwich, Melvin Johnson, Behnam Neyshabur, Justin Mao-Jones, Renshen Wang, Vinay Ramasesh, Zaheer Abbas, Arthur Guez, Constant Segal, Duc Dung Nguyen, James Svensson, Le Hou, Sarah York, Kieran Milan, Sophie Bridgers, Wiktor Gworek, Marco Tagliasacchi, James Lee-Thorp, Michael Chang, Alexey Guseynov, Ale Jakse Hartman, Michael Kwong, Ruizhe Zhao, Sheleem Kashem, Elizabeth Cole, Antoine Miech, Richard Tanburn, Mary Phuong, Filip Pavetic, Sebastien Cevey, Ramona Comanescu, Richard Ives, Sherry Yang, Cosmo Du, Bo Li, Zizhao Zhang, Mariko Iinuma, Clara Huiyi Hu, Aurko Roy, Shaan Bijwadia, Zhenkai Zhu, Danilo Martins, Rachel Saputro, Anita Gergely, Steven Zheng, Dawei Jia, Ioannis Antonoglou, Adam Sadovsky, Shane Gu, Yingying Bi, Alek Andreev, Sina Samangooei, Mina Khan, Tomas Kocisky, Angelos Filos, Chintu Kumar, Colton Bishop, Adams Yu, Sarah Hodkinson, Sid Mittal, Premal Shah, Alexandre Moufarek, Yong Cheng, Adam Bloniarz, Jaehoon Lee, Pedram Pejman, Paul Michel, Stephen Spencer, Vladimir Feinberg, Xuehan Xiong, Nikolay Savinov, Charlotte Smith, Siamak Shakeri, Dustin Tran, Mary Chesus, Bernd Bohnet, George Tucker, Tamara von Glehn, Carrie Muir, Yiran Mao, Hideto Kazawa, Ambrose Slone, Kedar Soparkar, Disha Shrivastava, James Cobon-Kerr, Michael Sharman, Jay Pavagadhi, Carlos Araya, Karolis Misiunas, Nimesh Ghelani, Michael Laskin, David Barker, Qiujia Li, Anton Briukhov, Neil Houlsby, Mia Glaese, Balaji Lakshminarayanan, Nathan Schucher, Yunhao Tang, Eli Collins, Hyeontaek Lim, Fangxiaoyu Feng, Adria Recasens, Guangda Lai, Alberto Magni, Nicola De Cao, Aditya Siddhant, Zoe Ashwood, Jordi Orbay, Mostafa Dehghani, Jenny Brennan, Yifan He, Kelvin Xu, Yang Gao, Carl Saroufim, James Molloy, Xinyi Wu, Seb Arnold, Solomon Chang, Julian Schrittwieser, Elena Buchatskaya, Soroush Radpour, Martin Polacek, Skye Giordano, Ankur Bapna, Simon Tokumine, Vincent Hellendoorn, Thibault Sottiaux, Sarah Cogan, Aliaksei Severyn, Mohammad Saleh, Shantanu Thakoor, Laurent Shefey, Siyuan Qiao, Meenu Gaba, Shuo yiin Chang, Craig Swanson, Biao Zhang, Benjamin Lee, Paul Kishan Rubenstein, Gan Song, Tom Kwiatkowski, Anna Koop, Ajay Kannan, David Kao, Parker Schuh, Axel Stjerngren, Golnaz Ghiasi, Gena Gibson, Luke Vilnis, Ye Yuan, Felipe Tiengo Ferreira, Aishwarya Kamath, Ted Klimenko, Ken Franko, Kefan Xiao, Indro Bhattacharya, Miteyan Patel, Rui Wang, Alex Morris, Robin Strudel, Vivek Sharma, Peter Choy, Sayed Hadi Hashemi, Jessica Landon, Mara Finkelstein, Priya Jhakra, Justin Frye, Megan Barnes, Matthew Mauger, Dennis Daun, Khuslen Baatarsukh, Matthew Tung, Wael Farhan, Henryk Michalewski, Fabio Viola, Felix de Chaumont Quitry, Charline Le Lan, Tom Hudson, Qingze Wang, Felix Fischer, Ivy Zheng, Elspeth White, Anca Dragan, Jean baptiste Alayrac, Eric Ni, Alexander Pritzel, Adam Iwanicki, Michael Isard, Anna Bulanova, Lukas Zilka, Ethan Dyer, Devendra Sachan, Srivatsan Srinivasan, Hannah Muckenhirn, Honglong Cai, Amol Mandhane, Mukarram Tariq, Jack W. Rae, Gary Wang, Kareem Ayoub, Nicholas FitzGerald, Yao Zhao, Woohyun Han, Chris Alberti, Dan Garrette, Kashyap Krishnakumar, Mai Gimenez, Anselm Levskaya, Daniel Sohn, Josip Matak, Inaki Iturrate, Michael B. Chang, Jackie Xiang, Yuan Cao, Nishant Ranka, Geoff Brown, Adrian Hutter, Vahab Mirrokni, Nanxin Chen, Kaisheng Yao, Zoltan Egyed, Francois Galilee, Tyler Liechty, Praveen Kallakuri, Evan Palmer, Sanjay Ghemawat, Jasmine Liu, David Tao, Chloe Thornton, Tim Green, Mimi Jasarevic, Sharon Lin, Victor Cotruta, Yi-Xuan Tan, Noah Fiedel, Hongkun Yu, Ed Chi, Alexander Neitz, Jens Heitkaemper, Anu Sinha, Denny Zhou, Yi Sun, Charbel Kaed, Brice Hulse, Swaroop Mishra, Maria Georgaki, Sneha Kudugunta, Clement Farabet, Izhak Shafran, Daniel Vlasic, Anton Tsitsulin, Rajagopal Ananthanarayanan, Alen Carin, Guolong Su, Pei Sun, Shashank V, Gabriel Carvajal, Josef Broder, Iulia Comsa, Alena Repina, William Wong, Warren Weilun Chen, Peter Hawkins, Egor Filonov, Lucia Loher, Christoph Hirnschall, Weiyi Wang, Jingchen Ye, Andrea Burns, Hardie Cate, Diana Gage Wright, Federico Piccinini, Lei Zhang, Chu-Cheng Lin, Ionel Gog, Yana Kulizhskaya, Ashwin Sreevatsa, Shuang Song, Luis C. Cobo, Anand Iyer, Chetan Tekur, Guillermo Garrido, Zhuyun Xiao, Rupert Kemp, Huaixiu Steven Zheng, Hui Li, Ananth Agarwal, Christel Ngani, Kati Goshvadi, Rebeca Santamaria-Fernandez, Wojciech Fica, Xinyun Chen, Chris Gorgolewski, Sean Sun, Roopal Garg, Xinyu Ye, S. M. Ali Eslami, Nan Hua, Jon Simon, Pratik Joshi, Yelin Kim, Ian Tenney, Sahitya Potluri, Lam Nguyen Thiet, Quan Yuan, Florian Luisier, Alexandra Chronopoulou, Salvatore Scellato, Praveen Srinivasan, Minmin Chen, Vinod Koverkathu, Valentin Dalibard, Yaming Xu, Brennan Saeta, Keith Anderson, Thibault Sellam, Nick Fernando, Fantine Huot, Junehyuk Jung, Mani Varadarajan, Michael Quinn, Amit Raul, Maigo Le, Ruslan Habalov, Jon Clark, Komal Jalan, Kalesha Bullard, Achintya Singhal, Thang Luong, Boyu Wang, Sujeevan Rajayogam, Julian Eisenschlos, Johnson Jia, Daniel Finchelstein, Alex Yakubovich, Daniel Balle, Michael Fink, Sameer Agarwal, Jing Li, Dj Dvijotham, Shalini Pal, Kai Kang, Jaclyn Konzelmann, Jennifer Beattie, Olivier Dousse, Diane Wu, Remi Crocker, Chen Elkind, Siddhartha Reddy Jonnalagadda, Jong Lee, Dan Holtmann-Rice, Krystal Kallarackal, Rosanne Liu, Denis Vnukov, Neera Vats, Luca Invernizzi, Mohsen Jafari, Huanjie Zhou, Lilly Taylor, Jennifer Prendki, Marcus Wu, Tom Eccles, Tianqi Liu, Kavya Kopparapu, Francoise Beaufays, Christof Angermueller, Andreea Marzoca, Shourya Sarcar, Hilal Dib, Jeff Stanway, Frank Perbet, Nejc Trdin, Rachel Sterneck, Andrey Khorlin, Dinghua Li, Xihui Wu, Sonam Goenka, David Madras, Sasha Goldshtein, Willi Gierke, Tong Zhou, Yaxin Liu, Yannie Liang, Anais White, Yunjie Li, Shreya Singh, Sanaz Bahargam, Mark Epstein, Sujoy Basu, Li Lao, Adnan Ozturel, Carl Crous, Alex Zhai, Han Lu, Zora Tung, Neeraj Gaur, Alanna Walton, Lucas Dixon, Ming Zhang, Amir Globerson, Grant Uy, Andrew Bolt, Olivia Wiles, Milad Nasr, Ilia Shumailov, Marco Selvi, Francesco Piccinno, Ricardo Aguilar, Sara McCarthy, Misha Khalman, Mrinal Shukla, Vlado Galic, John Carpenter, Kevin Villela, Haibin Zhang, Harry Richardson, James Martens, Matko Bosnjak, Shreyas Rammohan Belle, Jeff Seibert, Mahmoud Alnahlawi, Brian McWilliams, Sankalp Singh, Annie Louis, Wen Ding, Dan Popovici, Lenin Simicich, Laura Knight, Pulkit Mehta, Nishesh Gupta, Chongyang Shi, Saaber Fatehi, Jovana Mitrovic, Alex Grills, Joseph Pagadora, Tsendsuren Munkhdalai, Dessie Petrova, Danielle Eisenbud, Zhishuai Zhang, Damion Yates, Bhavishya Mittal, Nilesh Tripuraneni, Yannis Assael, Thomas Brovelli, Prateek Jain, Mihajlo Velimirovic, Canfer Akbulut, Jiaqi Mu, Wolfgang Macherey, Ravin Kumar, Jun Xu, Haroon Qureshi, Gheorghe Comanici, Jeremy Wiesner, Zhitao Gong, Anton Ruddock, Matthias Bauer, Nick Felt, Anirudh GP, Anurag Arnab, Dustin Zelle, Jonas Rothfuss, Bill Rosgen, Ashish Shenoy, Bryan Seybold, Xinjian Li, Jayaram Mudigonda, Goker Erdogan, Jiawei Xia, Jiri Simsa, Andrea Michi, Yi Yao, Christopher Yew, Steven Kan, Isaac Caswell, Carey Radebaugh, Andre Elisseeff, Pedro Valenzuela, Kay McKinney, Kim Paterson, Albert Cui, Eri Latorre-Chimoto, Solomon Kim, William Zeng, Ken Durden, Priya Ponnapalli, Tiberiu Sosea, Christopher A. Choquette-Choo, James Manyika, Brona Robenek, Harsha Vashisht, Sebastien Pereira, Hoi Lam, Marko Velic, Denese Owusu-Afriyie, Katherine Lee, Tolga Bolukbasi, Alicia Parrish, Shawn Lu, Jane Park, Balaji Venkatraman, Alice Talbert, Lambert Rosique, Yuchung Cheng, Andrei Sozanschi, Adam Paszke, Praveen Kumar, Jessica Austin, Lu Li, Khalid Salama, Bartek Perz, Wooyeol Kim, Nandita Dukkipati, Anthony Baryshnikov, Christos Kaplanis, XiangHai Sheng, Yuri Chervonyi, Caglar Unlu, Diego de Las Casas, Harry Askham, Kathryn Tunyasuvunakool, Felix Gimeno, Siim Poder, Chester Kwak, Matt Miecnikowski, Vahab Mirrokni, Alek Dimitriev, Aaron Parisi, Dangyi Liu, Tomy Tsai, Toby Shevlane, Christina Kouridi, Drew Garmon, Adrian Goedeckemeyer, Adam R. Brown, Anitha Vijayakumar, Ali Elqursh, Sadegh Jazayeri, Jin Huang, Sara Mc Carthy, Jay Hoover, Lucy Kim, Sandeep Kumar, Wei Chen, Courtney Biles, Garrett Bingham, Evan Rosen, Lisa Wang, Qijun Tan, David Engel, Francesco Pongetti, Dario de Cesare, Dongseong Hwang, Lily Yu, Jennifer Pullman, Srini Narayanan, Kyle Levin, Siddharth Gopal, Megan Li, Asaf Aharoni, Trieu Trinh, Jessica Lo, Norman Casagrande, Roopali Vij, Loic Matthey, Bramandia Ramadhana, Austin Matthews, CJ Carey, Matthew Johnson, Kremena Goranova, Rohin Shah, Shereen Ashraf, Kingshuk Dasgupta, Rasmus Larsen, Yicheng Wang, Manish Reddy Vuyyuru, Chong Jiang, Joana Ijazi, Kazuki Osawa, Celine Smith, Ramya Sree Boppana, Taylan Bilal, Yuma Koizumi, Ying Xu, Yasemin Altun, Nir Shabat, Ben Bariach, Alex Korchemniy, Kiam Choo, Olaf Ronneberger, Chimezie Iwuanyanwu, Shubin Zhao, David Soergel, Cho-Jui Hsieh, Irene Cai, Shariq Iqbal, Martin Sundermeyer, Zhe Chen, Elie Bursztein, Chaitanya Malaviya, Fadi Biadsy, Prakash Shroff, Inderjit Dhillon, Tejasi Latkar, Chris Dyer, Hannah Forbes, Massimo Nicosia, Vitaly Nikolaev, Somer Greene, Marin Georgiev, Pidong Wang, Nina Martin, Hanie Sedghi, John Zhang, Praseem Banzal, Doug Fritz, Vikram Rao, Xuezhi Wang, Jiageng Zhang, Viorica Patraucean, Dayou Du, Igor Mordatch, Ivan Jurin, Lewis Liu, Ayush Dubey, Abhi Mohan, Janek Nowakowski, Vlad-Doru Ion, Nan Wei, Reiko Tojo, Maria Abi Raad, Drew A. Hudson, Vaishakh Keshava, Shubham Agrawal, Kevin Ramirez, Zhichun Wu, Hoang Nguyen, Ji Liu, Madhavi Sewak, Bryce Petrini, DongHyun Choi, Ivan Philips, Ziyue Wang, Ioana Bica, Ankush Garg, Jarek Wilkiewicz, Priyanka Agrawal, Xiaowei Li, Danhao Guo, Emily Xue, Naseer Shaik, Andrew Leach, Sadh MNM Khan, Julia Wiesinger, Sammy Jerome, Abhishek Chakladar, Alek Wenjiao Wang,

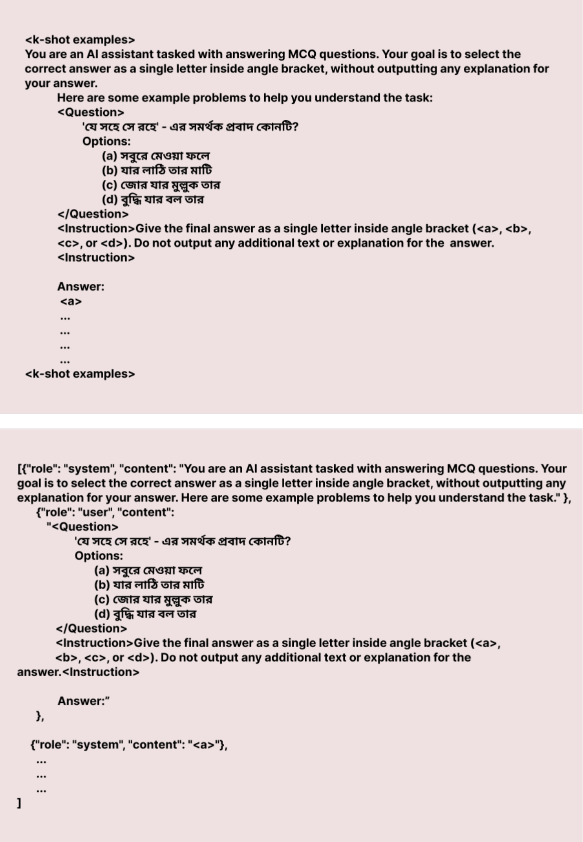
Tina Ornduff, Folake Abu, Alireza Ghaffarkhah, Marcus Wainwright, Mario Cortes, Frederick Liu, Joshua Maynez, Andreas Terzis, Pouya Samangouei, Riham Mansour, Tomasz Kępa, François-Xavier Aubet, Anton Algymr, Dan Banica, Agoston Weisz, Andras Orban, Alexandre Senges, Ewa Andrejczuk, Mark Geller, Niccolo Dal Santo, Valentin Anklin, Majd Al Merey, Martin Baeuml, Trevor Strohman, Junwen Bai, Slav Petrov, Yonghui Wu, Demis Hassabis, Koray Kavukcuoglu, Jeff Dean, and Oriol Vinyals. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

7 Appendix

7.1 Evaluation Details

Since our MCQ questions are largely factual-based and do not require reasoning for most cases, we set the maximum output token length is set to 1024 for all experiments. This allows use to analyze responses from models during cases where models produce verbose responses, primarily in 0-shot setting, due to lack of guiding examples in 5-shot setting, despite being instructed in the prompt to produce only option ID as output. We set the decoding temperature to 0.2 to reduce randomness, however, as shown in (Renze and Guven, 2024), changing temperature from 0 to 1 do not have a significant performance change in LLMs.

For 5-shot setting, we randomly pick 5 questions from each category. Since we perform a meticulous categorization and double-inspection process, our randomly selected samples are generally good representations of the category.

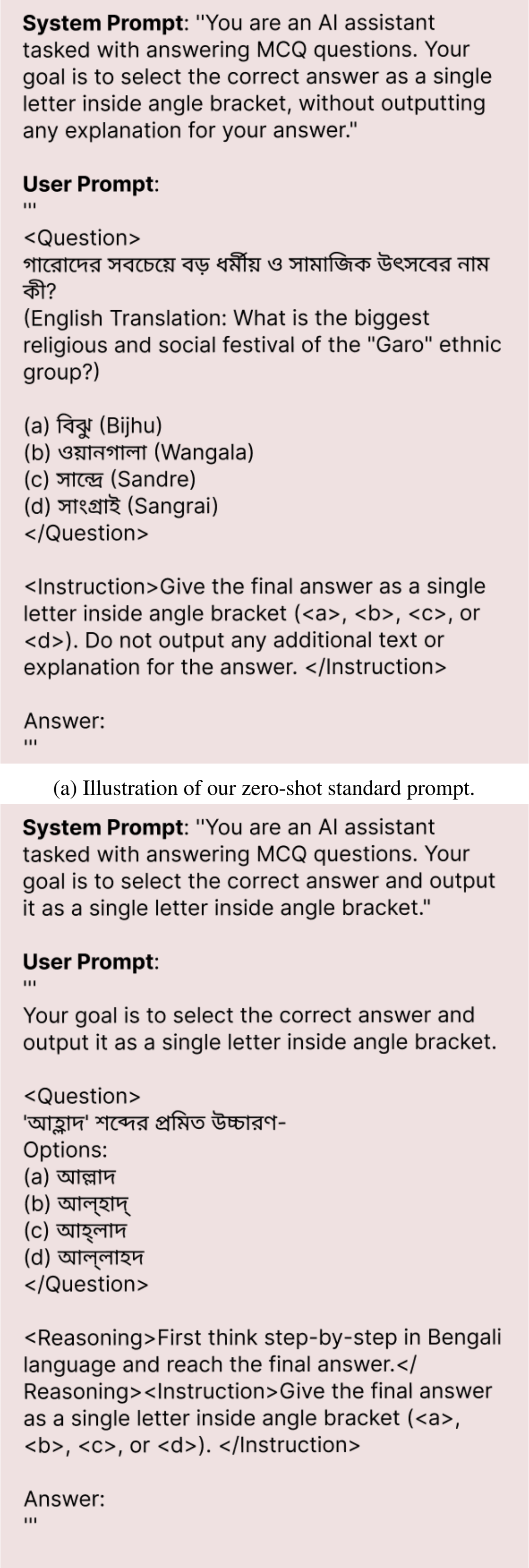


```
<k-shot examples>
You are an AI assistant tasked with answering MCQ questions. Your goal is to select the
correct answer as a single letter inside angle bracket, without outputting any explanation for
your answer.
        Here are some example problems to help you understand the task:
        <Question>
              'যে সহে সে রহে' - এর সমর্থক প্রবাদ কোনটি?
              Options:
                   (a) সবুরে মেওয়া ফলে
                   (b) যার লাঠি তার মাটি
                   (c) জোর যার মুল্লুক তার
                   (d) বুদ্ধি যার বল তার
        </Question>
        <Instruction>Give the final answer as a single letter inside angle bracket (<a>, <b>,
        <c>, or <d>). Do not output any additional text or explanation for the  answer.
        <Instruction>

        Answer:
         <a>
         ...
         ...
         ...
         ...
<k-shot examples>
```

```
[{"role": "system", "content": "You are an AI assistant tasked with answering MCQ questions. Your
goal is to select the correct answer as a single letter inside angle bracket, without outputting any
explanation for your answer. Here are some example problems to help you understand the task." },
     {"role": "user", "content":
       "<Question>
              'যে সহে সে রহে' - এর সমর্থক প্রবাদ কোনটি?
              Options:
                   (a) সবুরে মেওয়া ফলে
                   (b) যার লাঠি তার মাটি
                   (c) জোর যার মুল্লুক তার
                   (d) বুদ্ধি যার বল তার
          </Question>
          <Instruction>Give the final answer as a single letter inside angle bracket (<a>,
          <b>, <c>, or <d>). Do not output any additional text or explanation for the
answer.<Instruction>

          Answer:”
     },

    {"role": "system", "content": "<a>"},
     ...
     ...
     ...
]
```

Figure 2: Prompt Structure for 5-shot setting using GPT model.

7.2 Additional BLUCK Results



```
System Prompt: "You are an AI assistant
tasked with answering MCQ questions. Your
goal is to select the correct answer as a single
letter inside angle bracket, without outputting
any explanation for your answer."

User Prompt:
'''
<Question>
গারোদের সবচেয়ে বড় ধর্মীয় ও সামাজিক উৎসবের নাম
কী?
(English Translation: What is the biggest
religious and social festival of the "Garo" ethnic
group?)

(a) বিঝু (Bijhu)
(b) ওয়ানগালা (Wangala)
(c) সান্দ্রে (Sandre)
(d) সাংগ্রাই (Sangrai)
</Question>

<Instruction>Give the final answer as a single
letter inside angle bracket (<a>, <b>, <c>, or
<d>). Do not output any additional text or
explanation for the answer. </Instruction>

Answer:
'''
```

(a) Illustration of our zero-shot standard prompt.

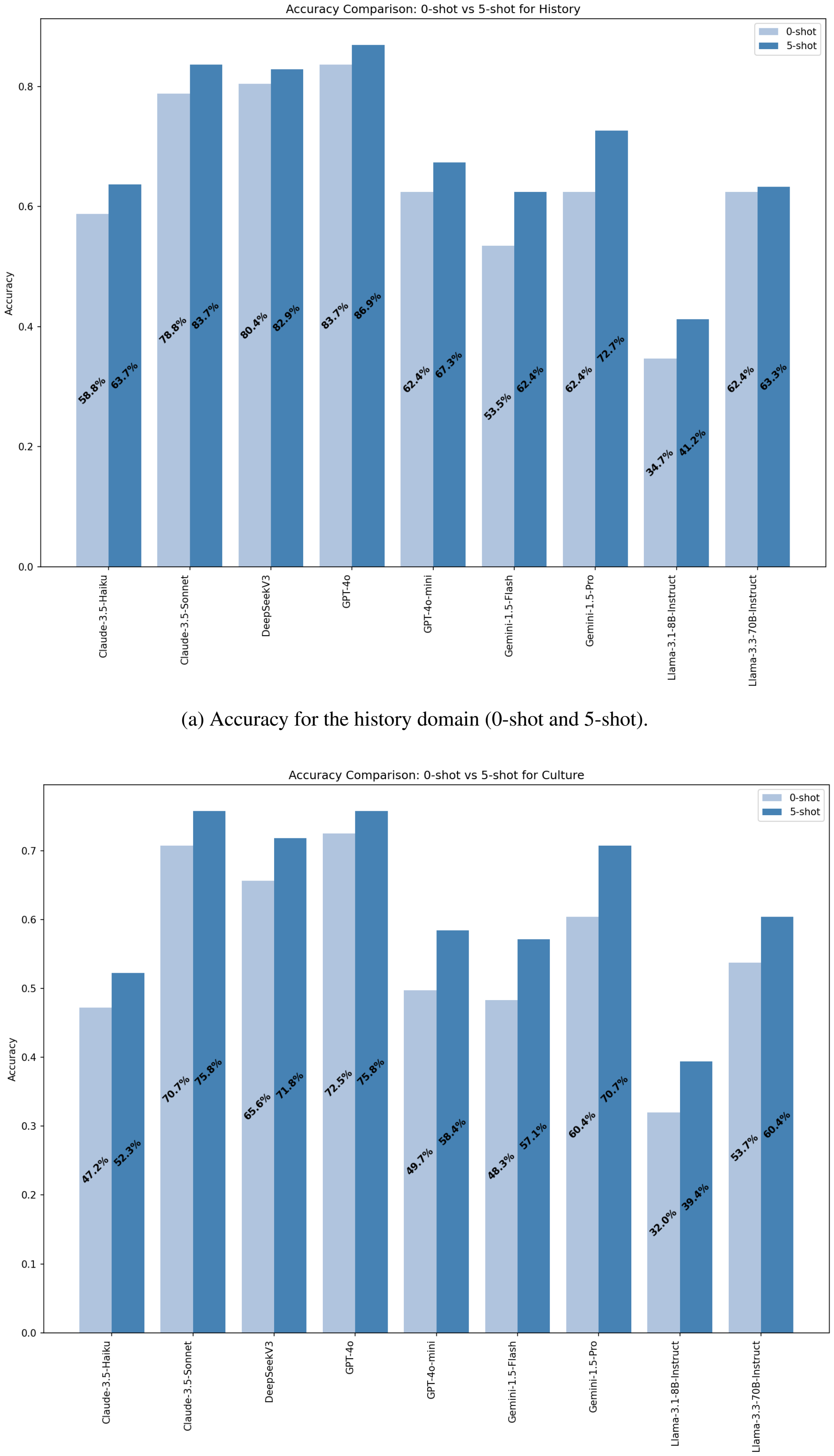
```
System Prompt: "You are an AI assistant
tasked with answering MCQ questions. Your
goal is to select the correct answer and output
it as a single letter inside angle bracket."

User Prompt:
'''
Your goal is to select the correct answer and
output it as a single letter inside angle bracket.

<Question>
'আহ্লাদ' শব্দের প্রমিত উচ্চারণ-
Options:
(a) আল্লাদ
(b) আল্হাদ্
(c) আহ্লাদ
(d) আল্লাহদ
</Question>

<Reasoning>First think step-by-step in Bengali
language and reach the final answer.</
Reasoning><Instruction>Give the final answer
as a single letter inside angle bracket (<a>,
<b>, <c>, or <d>). </Instruction>

Answer:
'''
```

(b) Prompt of zero-shot CoT reasoning by Claude-3.5-Sonnet on the Pronunciation category.

Figure 3: Prompting strategies used in our evaluation.

Categories		GPT-4o-mini		Claude-3.5-Haiku		Gemini-1.5-Flash		Llama-3.1-8B	
		0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot
History	Ancient Bengal	0.667	**0.737**	0.687	0.717	0.646	0.697	0.394	0.404
	British Bengal	0.775	0.8	0.7	**0.825**	0.675	0.8	0.35	0.475
	Pakistan Era	0.528	**0.566**	0.453	0.491	0.377	0.491	0.302	0.396
	Average	0.624	**0.673**	0.588	0.637	0.535	0.624	0.347	0.412
Culture	Indigenous People	0.484	**0.677**	0.452	0.645	0.355	0.581	0.355	0.452
	Arts, Heritage & Media	0.478	**0.536**	0.42	0.478	0.449	0.449	0.29	0.377
	National Issues	0.4	0.533	0.267	0.4	0.467	**0.667**	0.133	0.533
	Constitution	0.677	0.774	0.581	0.71	0.645	**0.839**	0.355	0.387
	Resources	0.472	**0.722**	0.417	0.5	0.472	0.583	0.25	0.389
	Geography	0.563	**0.598**	0.46	0.506	0.471	0.529	0.299	0.333
	Law	0.472	0.546	0.496	0.514	0.493	**0.577**	0.345	0.405
	Average	0.497	**0.584**	0.472	0.523	0.483	0.571	0.320	0.394
Phonetics	Alphabet	0.2	0.8	0.7	0.8	0.7	**0.9**	0.2	0.7
	Pronunciation	0.159	0.275	0.261	0.275	0.203	**0.319**	0.217	0.29
	Conjunct Letters	0.478	0.652	0.783	**0.957**	0.609	0.783	0.478	0.522
	Sound & Letters	0.5	0.625	0.438	0.646	0.625	**0.667**	0.25	0.292
	Sound Changes	0.278	0.333	0.315	0.444	0.278	**0.481**	0.296	0.352
	Phonetic Combining Rules	0.402	0.418	0.435	**0.505**	0.457	0.478	0.31	0.359
	Miscellaneous Phonetics	0.55	0.6	0.525	**0.613**	0.575	0.575	0.4	0.363
	Average	0.387	0.459	0.434	**0.526**	0.449	0.515	0.310	0.357
Semantics	Synonyms	0.681	0.747	0.761	**0.843**	0.687	0.775	0.385	0.426
	Antonyms	0.642	0.691	0.679	**0.77**	0.691	0.758	0.412	0.527
	One Word Expressions	0.506	0.567	0.589	**0.661**	0.522	0.606	0.433	0.406
	Idioms	0.515	0.5	0.444	0.495	0.48	**0.581**	0.354	0.333
	Proverbs	0.66	0.66	0.638	0.723	0.66	**0.787**	0.404	0.426
	Miscellaneous	**0.616**	0.589	0.521	0.568	0.514	0.555	0.363	0.39
	Average	0.607	0.640	0.626	**0.698**	0.599	0.681	0.389	0.416
Overall Average		0.540	0.595	0.548	**0.617**	0.536	**0.617**	0.353	0.399

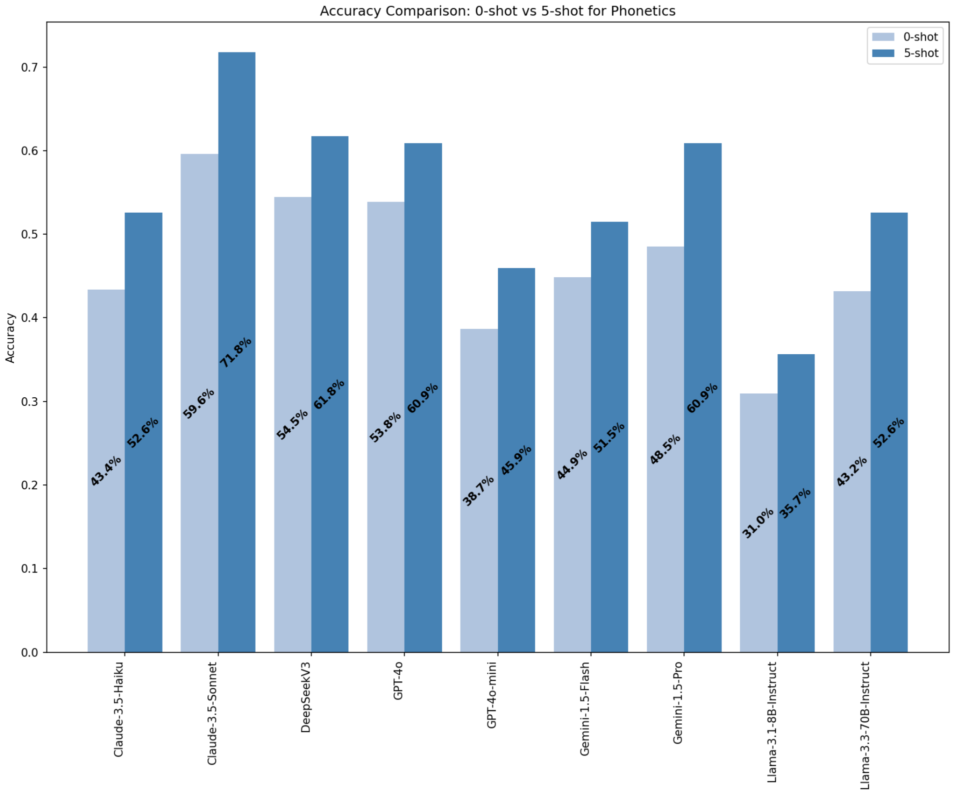
Table 4: BLUCK benchmark comparison by subcategories and major categories across smaller models in 0-shot and 5-shot settings. The highest accuracy(s) for each category are boldy marked.



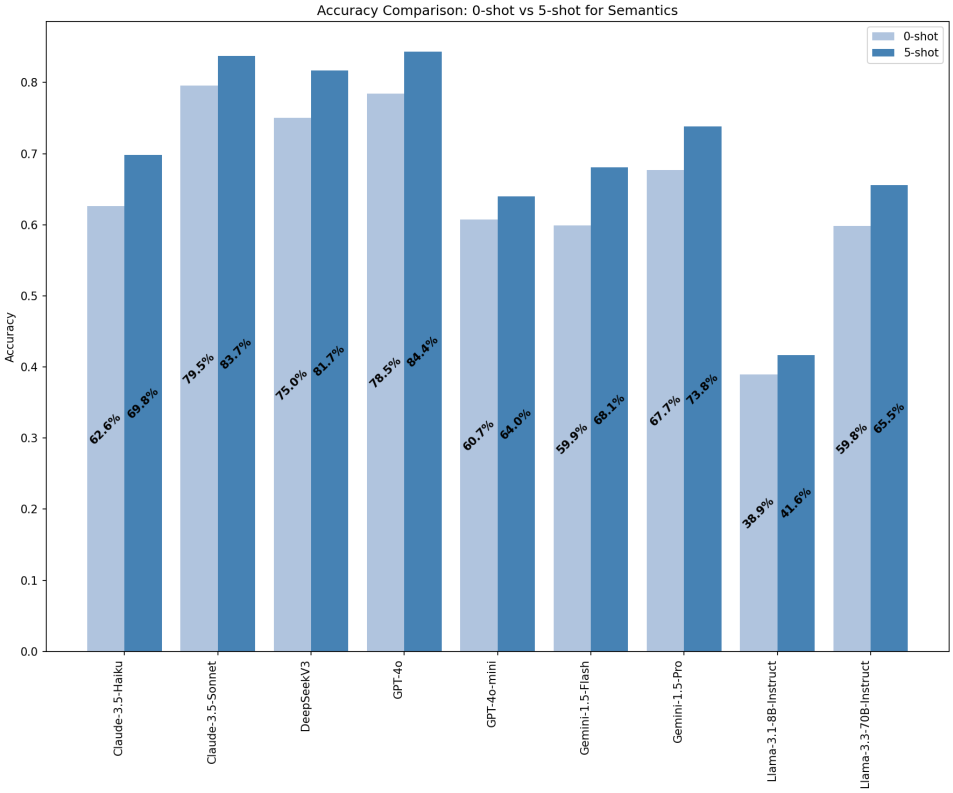
(a) Accuracy for the history domain (0-shot and 5-shot).

(b) Accuracy for the culture domain (0-shot and 5-shot).

Figure 4: Comparison of accuracy across history and culture domains under 0-shot and 5-shot settings.

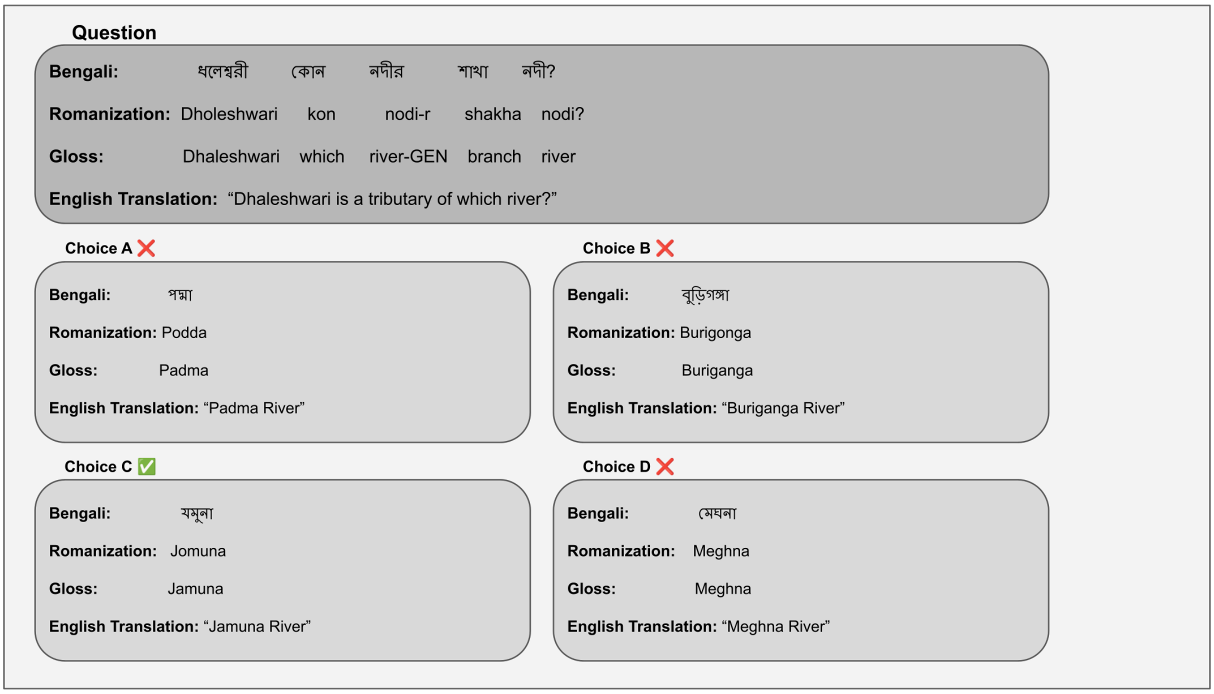


(a) Accuracy for the phonetics domain (0-shot and 5-shot).



(b) Accuracy for the semantics domain (0-shot and 5-shot).

Figure 5: Comparison of accuracy across phonetics and semantics domains under 0-shot and 5-shot settings.



Question

Bengali:	ধলেশ্বরী	কোন	নদীর	শাখা	নদী?
Romanization:	Dholeshwari	kon	nodi-r	shakha	nodi?
Gloss:	Dhaleshwari	which	river-GEN	branch	river

English Translation: “Dhaleshwari is a tributary of which river?”

Choice A ❌

Bengali: পদ্মা

Romanization: Podda

Gloss: Padma

English Translation: “Padma River”

Choice B ❌

Bengali: বুড়িগঙ্গা

Romanization: Burigonga

Gloss: Buriganga

English Translation: “Buriganga River”

Choice C ✅

Bengali: যমুনা

Romanization: Jomuna

Gloss: Jamuna

English Translation: “Jamuna River”

Choice D ❌

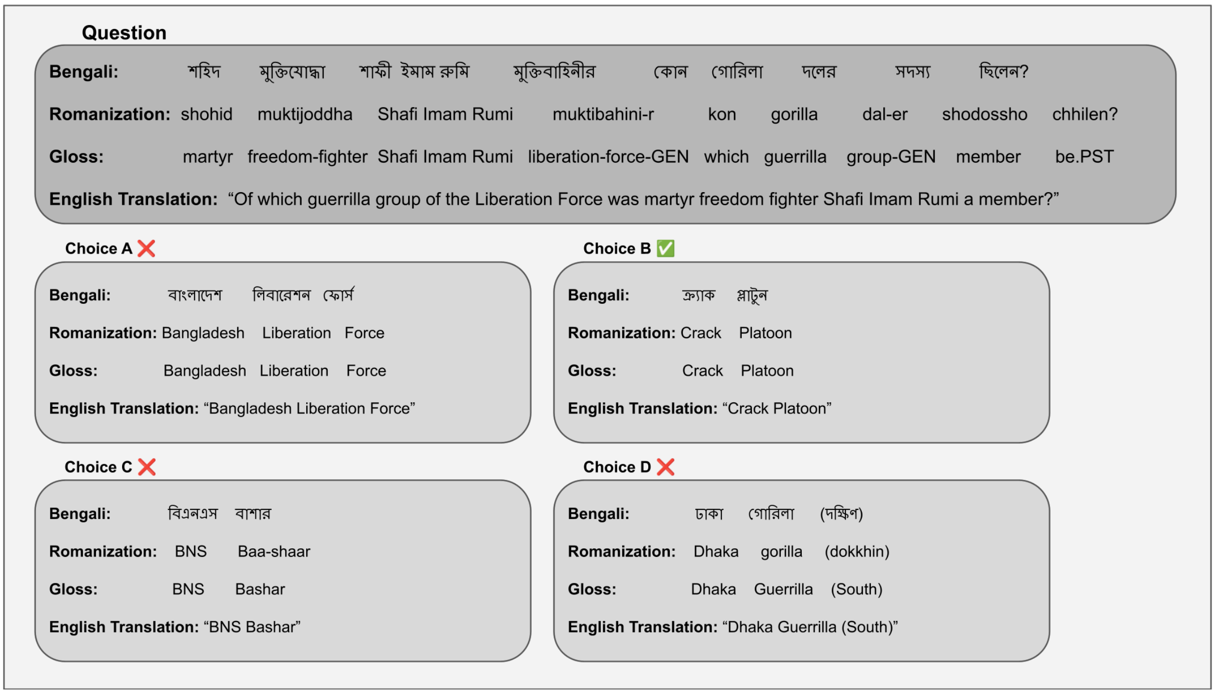
Bengali: মেঘনা

Romanization: Meghna

Gloss: Meghna

English Translation: “Meghna River”

Figure 6: Example MCQA from the Culture domain.



Question

Bengali:	শহিদ	মুক্তিযোদ্ধা	শাফী ইমাম রুমি	মুক্তিবাহিনীর	কোন	গোরিলা	দলের	সদস্য	ছিলেন?
Romanization:	shohid	muktijoddha	Shafi Imam Rumi	muktibahini-r	kon	gorilla	dal-er	shodossho	chhilen?
Gloss:	martyr	freedom-fighter	Shafi Imam Rumi	liberation-force-GEN	which	guerrilla	group-GEN	member	be.PST

English Translation: “Of which guerrilla group of the Liberation Force was martyr freedom fighter Shafi Imam Rumi a member?”

Choice A ❌

Bengali:	বাংলাদেশ	লিবারেশন	ফোর্স
Romanization:	Bangladesh	Liberation	Force
Gloss:	Bangladesh	Liberation	Force

English Translation: “Bangladesh Liberation Force”

Choice B ✅

Bengali:	ক্র্যাক	প্লাটুন
Romanization:	Crack	Platoon
Gloss:	Crack	Platoon

English Translation: “Crack Platoon”

Choice C ❌

Bengali:	বিএনএস	বাশার
Romanization:	BNS	Baa-shaar
Gloss:	BNS	Bashar

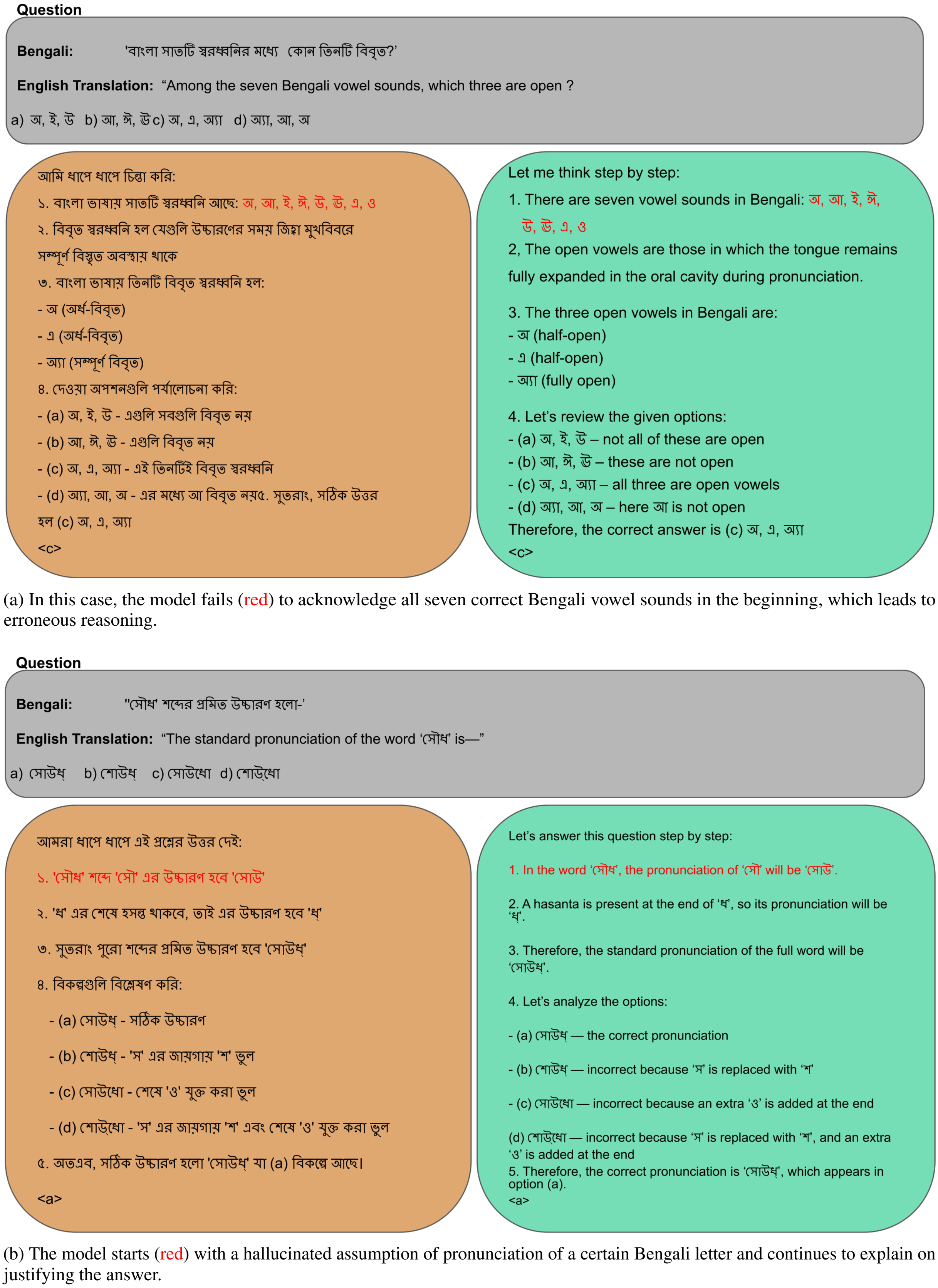
English Translation: “BNS Bashar”

Choice D ❌

Bengali:	ঢাকা	গোরিলা	(দক্ষিণ)
Romanization:	Dhaka	gorilla	(dokkhin)
Gloss:	Dhaka	Guerrilla	(South)

English Translation: “Dhaka Guerrilla (South)”

Figure 7: Example MCQA from the History domain.



Question

Bengali: 'বাংলা সাতটি স্বরধ্বনির মধ্যে কোন তিনটি বিবৃত?'

English Translation: “Among the seven Bengali vowel sounds, which three are open ?

a) অ, ই, উ b) আ, ঈ, ঊ c) অ, এ, অ্যা d) অ্যা, আ, অ

আমি ধাপে ধাপে চিন্তা করি:

১. বাংলা ভাষায় সাতটি স্বরধ্বনি আছে: অ, আ, ই, ঈ, উ, ঊ, এ, ও

২. বিবৃত স্বরধ্বনি হল যেগুলি উচ্চারণের সময় জিহ্বা মুখবিবরে সম্পূর্ণ বিস্তৃত অবস্থায় থাকে

৩. বাংলা ভাষায় তিনটি বিবৃত স্বরধ্বনি হল:

- অ (অর্ধ-বিবৃত)
- এ (অর্ধ-বিবৃত)
- অ্যা (সম্পূর্ণ বিবৃত)

৪. দেওয়া অপশনগুলি পর্যালোচনা করি:

- (a) অ, ই, উ - এগুলি সবগুলি বিবৃত নয়
- (b) আ, ঈ, ঊ - এগুলি বিবৃত নয়
- (c) অ, এ, অ্যা - এই তিনটিই বিবৃত স্বরধ্বনি
- (d) অ্যা, আ, অ - এর মধ্যে আ বিবৃত নয়৫. সুতরাং, সঠিক উত্তর হল (c) অ, এ, অ্যা

<c>

Let me think step by step:

1. There are seven vowel sounds in Bengali: অ, আ, ই, ঈ, উ, ঊ, এ, ও

2, The open vowels are those in which the tongue remains fully expanded in the oral cavity during pronunciation.

3. The three open vowels in Bengali are:
- অ (half-open)
- এ (half-open)
- অ্যা (fully open)

4. Let’s review the given options:
- (a) অ, ই, উ – not all of these are open
- (b) আ, ঈ, ঊ – these are not open
- (c) অ, এ, অ্যা – all three are open vowels
- (d) অ্যা, আ, অ – here আ is not open

Therefore, the correct answer is (c) অ, এ, অ্যা

<c>

(a) In this case, the model fails (red) to acknowledge all seven correct Bengali vowel sounds in the beginning, which leads to erroneous reasoning.

Question

Bengali: "সৌধ' শব্দের প্রমিত উচ্চারণ হলো-'

English Translation: “The standard pronunciation of the word ‘সৌধ’ is—”

a) সোউধ্ b) শোউধ্ c) সোউধো d) শোউ্ধো

আমরা ধাপে ধাপে এই প্রশ্নের উত্তর দেই:

১. 'সৌধ' শব্দে 'সৌ' এর উচ্চারণ হবে 'সোউ'

২. 'ধ' এর শেষে হসন্ত থাকবে, তাই এর উচ্চারণ হবে 'ধ্'

৩. সুতরাং পুরো শব্দের প্রমিত উচ্চারণ হবে 'সোউধ্'

৪. বিকল্পগুলি বিশ্লেষণ করি:

- (a) সোউধ্ - সঠিক উচ্চারণ
- (b) শোউধ্ - 'স' এর জায়গায় 'শ' ভুল
- (c) সোউধো - শেষে 'ও' যুক্ত করা ভুল
- (d) শোউ্ধো - 'স' এর জায়গায় 'শ' এবং শেষে 'ও' যুক্ত করা ভুল

৫. অতএব, সঠিক উচ্চারণ হলো 'সোউধ্' যা (a) বিকল্পে আছে।

<a>

Let’s answer this question step by step:

1. In the word ‘সৌধ’, the pronunciation of ‘সৌ’ will be ‘সোউ’.

2. A hasanta is present at the end of ‘ধ’, so its pronunciation will be ‘ধ্’.

3. Therefore, the standard pronunciation of the full word will be ‘সোউধ্’.

4. Let’s analyze the options:

- (a) সোউধ্ — the correct pronunciation
- (b) শোউধ্ — incorrect because ‘স’ is replaced with ‘শ’
- (c) সোউধো — incorrect because an extra ‘ও’ is added at the end

(d) শোউ্ধো — incorrect because ‘স’ is replaced with ‘শ’, and an extra ‘ও’ is added at the end
5. Therefore, the correct pronunciation is ‘সোউধ্’, which appears in option (a).
<a>

(b) The model starts (red) with a hallucinated assumption of pronunciation of a certain Bengali letter and continues to explain on justifying the answer.

Figure 8: Failure Case in CoT Reasoning for Pronunciation Subcategory

BanHateMe : Understanding Hate in Bangla Memes through Detection, Categorization, and Target Profiling

Md Ayon Mia[1], **Md Fahim**[2,3]

[1]*Dhaka International University*
[2]*Center for Computational & Data Sciences* [3]*Penta Global Limited*
Correspondence: {mdayonrahman100, fahimcse381}@gmail.com

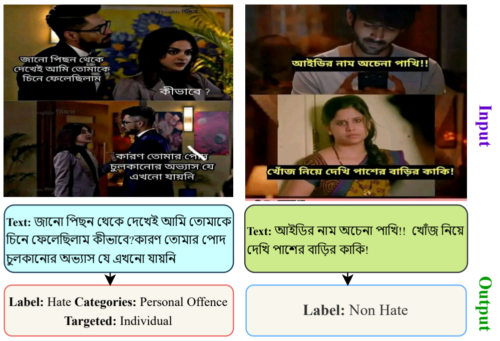
Abstract

Detecting hateful memes is a complex task due to the interplay of text and visuals, with subtle cultural cues often determining whether content is harmful. This challenge is amplified in Bangla, a low-resource language where existing resources provide only binary labels or single dimensions of hate. To bridge this gap, we introduce BanHateMe , a comprehensive Bangla hateful meme dataset with hierarchical annotations across three levels: binary hate, hate categories, and targeted groups. The dataset comprises 3,819 culturally grounded memes, annotated with substantial inter-annotator agreement. We further propose a hierarchical loss function that balances predictions across levels, preventing bias toward binary detection at the expense of fine-grained classification. To assess performance, we pair pretrained language and vision models and systematically evaluate three multimodal fusion strategies: summation, concatenation, and co-attention, demonstrating the effectiveness of hierarchical learning and cross-modal alignment. Our work establishes BanHateMe as a foundational resource for fine-grained multimodal hate detection in Bangla and contributes key insights for content moderation in low-resource settings. We release the code and dataset publicly at https://github.com/Ayon128/BanHateMe.

Disclaimer: This paper includes examples that may be offensive. Such content is presented only for research purposes and is unavoidable given the nature of the study.

1 Introduction

Memes have rapidly become one of the most influential forms of online communication, combining images with short text to convey ideas, humor, and social commentary. While often entertaining, they are also frequently used to spread hate in subtle and multimodal ways, embedding socio-political cues, cultural references, or stereotypes that can harm individuals and groups based on gender, politics, or religion. Their multimodal nature and concealed semantics make them especially difficult to analyze, as harmful meaning often arises from the interaction between visual and textual elements (Zannettou et al., 2018; Kiela et al., 2020). Moreover, meme content evolves dynamically with changing events, metaphors, and linguistic patterns, complicating detection even further (Pramanick et al., 2021). These challenges are particularly acute in low-resource languages such as Bangla, where multimodal hate has received little attention despite the language's widespread use online.


জানো পিছন থেকে দেখেই আমি তোমাকে চিনে ফেলেছিলাম
কীভাবে ?
কারণ তোমার পোঁদ চুলকানোর অভ্যাস যে এখনো যায়নি
আইডির নাম অচেনা পাখি!!
খোঁজ নিয়ে দেখি পাশের বাড়ির কাকি!
Input
Output

Text: জানো পিছন থেকে দেখেই আমি তোমাকে চিনে ফেলেছিলাম কীভাবে?কারণ তোমার পোদ চুলকানোর অভ্যাস যে এখনো যায়নি

Label: Hate **Categories:** Personal Offence **Targeted:** Individual

Text: আইডির নাম অচেনা পাখি!! খোঁজ নিয়ে দেখি পাশের বাড়ির কাকি!

Label: Non Hate

Figure 1: Given a meme image with associated text as input, the output is its hierarchical annotation: hateful memes are labeled with a hate category and targeted group (left), while non-hateful memes are marked as non hate (right).

To effectively assess hateful memes, it is not enough to simply determine whether the content is hateful. A comprehensive understanding requires capturing both the severity of hate through categories such as abusive, political, gender, personal offence, or religious, and the intended targets, including individuals, communities, organizations, or society. Existing Bangla meme datasets fall short in this regard: MUTE (Hossain et al., 2022)

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 180–195
December 23, 2025 ©2025 Association for Computational Linguistics

Dataset	Task Type	Hate Cat	Target Groups	#Samples
Hossain et al., 2024b	Hate/Non-Hate & Target Entity Detection	✗	✓	7,148
Ahsan et al., 2024	Aggression Detection	✓	✗	4,848
Hossain et al., 2022	Hate vs. Non-Hate	✗	✗	4,158
Das and Mukherjee, 2023	Abusive vs. Non-Abusive	✗	✗	4,043
Ours	Hierarchical Classification	✓	✓	3,819

Table 1: Comparison of Bengali multimodal meme datasets. BANHATEME introduces hierarchical annotations, including binary labels, hate categories, and target groups, which are not jointly supported in prior resources.

provides only binary labels, BanglaAbuseMeme (Das and Mukherjee, 2023) focuses solely on abusive content, while more recent efforts such as BHM (Hossain et al., 2024b) and MIMOSA (Ahsan et al., 2024) annotate either categories or targets but never both. As summarized in Table 1, no current resource jointly supports all levels of analysis. To fill this gap, we introduce BANHATEME , a comprehensive Bangla multimodal hateful meme dataset with hierarchical annotation, where memes are classified as hateful or not, and hateful memes are further categorized by type and targeted group, thereby bridging this gap, as illustrated in Figure 1. We complement this design with a hierarchical loss function that balances predictions across levels, ensuring performance is not skewed toward binary detection at the expense of fine-grained recognition. Since Bangla lacks dedicated vision–language models, we combine pretrained language and vision encoders and systematically evaluate three fusion strategies—summation, concatenation, and co-attention to address alignment challenges. Our experiments show that BanglaBERT with Swin Transformer and concatenation yields competitive results, while co-attention provides clear improvements in target group recognition. These findings underscore the importance of hierarchical modeling and cross-modal alignment for multimodal hate detection in Bangla. Our key contributions are as follows:

- We introduce BANHATEME , the first Bangla hateful meme dataset with hierarchical annotations across binary labels, hate categories, and targeted groups.
- Propose a hierarchical loss function to balance supervision across multiple levels.
- Conduct a comprehensive evaluation comparing three multimodal fusion strategies, demonstrating the effectiveness of hierarchical learning for Bangla multimodal hate detection.

2 Related Work

2.1 Hateful memes dataset.

The release of the Hateful Memes Challenge (Kiela et al., 2020) established a benchmark for multimodal hate detection, highlighting the need for joint reasoning across text and images. Since then, several English datasets have expanded the space, including large-scale resources for offensive or harmful memes (Suryawanshi et al., 2020; Gomez et al., 2020; Pramanick et al., 2021). Efforts have also extended to low-resource languages, such as Hindi (Kumari et al., 2023; Rajput et al., 2022) and Greek (Perifanos and Goutsos, 2021). For Bangla, multimodal resources remain limited. MUTE (Hossain et al., 2022) introduced 4,158 memes labeled for binary hate detection, while BanglaAbuseMeme (Das and Mukherjee, 2023) provided 4,043 abusive memes. More recently, BHM (Hossain et al., 2024b) added target entity annotations (e.g., Individual, Organization, Community, Society), and MIMOSA (Ahsan et al., 2024) focused on aggression-specific categories such as Political, Gender, and Religious. BANMIME (Mia et al., 2025) further contributed to the Bangla multimodal landscape by introducing 2,000 misogynistic memes annotated with metaphor localization and human-written explanations. Beyond these resources, ExMUTE (Debnath et al., 2025) expands Bangla multimodal hate research by incorporating contextual labels across religion, politics, gender, and other domains, demonstrating the importance of context-aware annotations for improving hateful meme understanding.

2.2 Hateful memes detection methods.

Research on multimodal hateful memes has explored a range of fusion techniques. Conventional fusion approaches concatenate text and image features to form a joint representation (Vijayaraghavan et al., 2021; Gomez et al., 2020). Some works adopted bilinear pooling (Chandra et al., 2021),

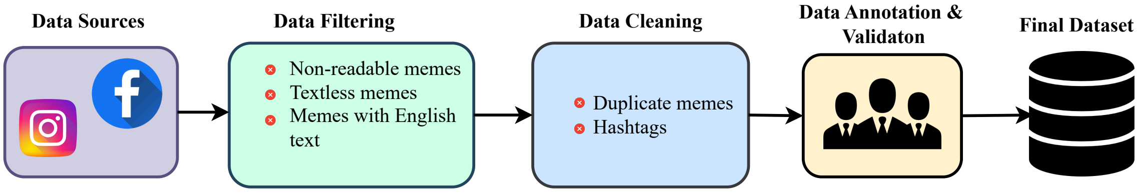


Figure 2: BanHateMe dataset development pipeline showing data collection from social media platforms, filtering to discard irrelevant content, cleaning to remove duplicates and extraneous elements, and annotation with validation to construct the final dataset.

while others fine-tuned vision–language transformer architectures such as ViLBERT, MMBT, and VisualBERT (Kiela et al., 2020). More recent studies explored prompting techniques for English hateful memes (Cao et al., 2023). Despite these advances, aligning textual and visual features remains underexplored, even though effective feature alignment is critical for robust multimodal representations (Zeng et al., 2021; Liu et al., 2019). In the Bangla context, emerging approaches such as Align-before-Attend (Hossain et al., 2024a) and Multimodal Attentive Fusion (Ahsan et al., 2024) demonstrate the value of improved cross-modal integration for meme classification.

3 BanHateMe : Dataset Creation

Following the limitations of existing Bangla hate meme datasets highlighted in Table 1, our dataset design focuses on enriching memes with layered annotations that move beyond binary hate labels (Hate vs. Non-Hate) to include five hate categories and four targeted group types. Particular attention was given to sourcing content from diverse platforms, capturing cultural and social nuances during annotation. The overall dataset construction pipeline is illustrated in Figure 2.

Data Collection. We collected memes from publicly accessible Bangla-speaking communities on major social media platforms, primarily Facebook and Instagram, between April 2022 and May 2025. We used search terms such as "Bangla memes", "Bangla hate memes", "Bangla abusive memes", "Bangla political memes", etc. to identify relevant meme sources. To comply with copyright and ethical standards, only memes from open groups, public pages, and non-private sources were included. In total, 5,560 memes were initially collected, with Facebook contributing 3,562 samples and Instagram providing the remaining 1,998.

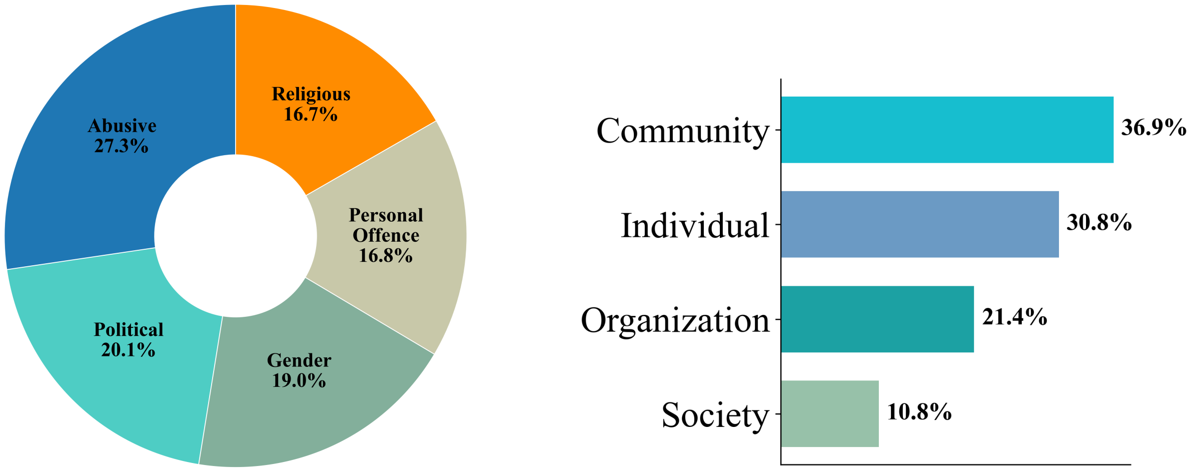
Data Filtering. To ensure the quality and relevance of our dataset, we applied a rigorous filtering procedure. Specifically, we removed: (1) memes with unreadable visual or textual content, (2) memes lacking any textual information, and (3) memes containing only English text, as our study emphasizes Bangla linguistic and cultural markers. This process resulted in the removal of 1,012 memes, leaving the 4,548 samples with extractable, readable Bangla text for subsequent analysis.

Data Cleaning. During this stage, we removed duplicate memes that appeared across different sources. We also stripped away non-informative textual elements, such as hashtags, which could add noise without contributing to the semantic analysis. After this cleaning procedure, 729 redundant and extraneous entries were discarded, leaving a final dataset of 3,819 text-bearing memes ready for annotation.

Text Extraction. Existing OCR systems perform poorly on Bangla text embedded in images, often producing noisy or incomplete outputs. To ensure accuracy, we relied on manual transcription of meme text. Two native typists were recruited to carry out the task, with the dataset evenly divided between them. Each typist was compensated at a rate of 1.5 BDT per sample.

Data Annotation. To annotate the BanHateMe dataset, we hired three Bangla-speaking undergraduate annotators with strong familiarity with local meme culture and online discourse. Their background in meme creation and cultural interpretation enabled them to recognize subtle hateful cues that might otherwise be overlooked. The annotators were provided with detailed annotation guidelines (Appendix Section A) and were instructed to complete the task within 25 days.

Annotation followed a two-stage hierarchical process. In the first stage, annotators determined whether each meme should be labeled as Hate or Not Hate. In the second stage, hateful memes were further classified into five hate categories—Abusive, Political, Gender, Personal Of-



(a) Hate Categories

(b) Targeted Groups

Figure 3: Distribution of hateful memes in the BanHateMe dataset across (a) hate categories and (b) targeted groups.

	Label	Kappa(κ)	Avg.
Primary	Hate	0.81	0.78
	Non Hate	0.75	
Hate Categories	Abusive	0.64	0.68
	Political	0.70	
	Personal	0.66	
	Gender	0.71	
	Religious	0.69	
Targeted Groups	Individual	0.71	0.69
	Organization	0.69	
	Community	0.68	
	Society	0.67	

Table 2: Inter-annotator agreement for the BanHateMe dataset, measured using Cohen's kappa (κ) across the binary task, hate categories, and targeted groups, with averages indicating substantial reliability.

fence, and Religious, followed by the study (Haider et al., 2024). Each hateful meme was also tagged with one of four targeted group types—Community, Individual, Organization, or Society, inspired by the work (Hossain et al., 2024b). All memes were annotated independently by three annotators, and final labels are assigned through majority voting to reduce individual bias. Annotators were compensated at a rate of 2 BDT per sample.

Data Validation. We assessed inter-annotator reliability using Cohen's kappa (κ) for each level of the hierarchical labeling task, with detailed results presented in Table 2. Overall, the scores indicate substantial agreement for the binary classification as well as across both hate categories and targeted groups. No score is less than 0.64 which indicates a good agreement between the annotators. The BanHateMe dataset serves as a trustworthy resource for Bangla memes.

Source Distribution	# Samples
Facebook	2517
Instagram	1302
Splits	
- Train	2673
- Val	381
- Test	765
Text Statistics	
Max Character Length	611
Mean Character Length	82.36
Min Character Length	10
Max Word Count	111
Mean Word Count	14.35
Min Word Count	3

Table 3: Statistical overview of the BanHateMe dataset, showing source distribution, data splits, and text statistics.

4 BanHateMe : Dataset Statistics

Meme Collection. The BanHateMe dataset consists of 3,819 labeled Bangla memes collected from two major platforms: Facebook (2,517) and Instagram (1,302). To enable systematic evaluation, we applied a stratified 70–10–20 split, resulting in training (2,673), validation (381), and test (765) partitions while preserving the overall distribution of labels. The dataset exhibits considerable linguistic diversity, with meme texts ranging from 10 to 611 characters and 3 to 111 words, averaging 82.36 characters and 14.35 words per instance. Table 3 summarizes the dataset statistics.

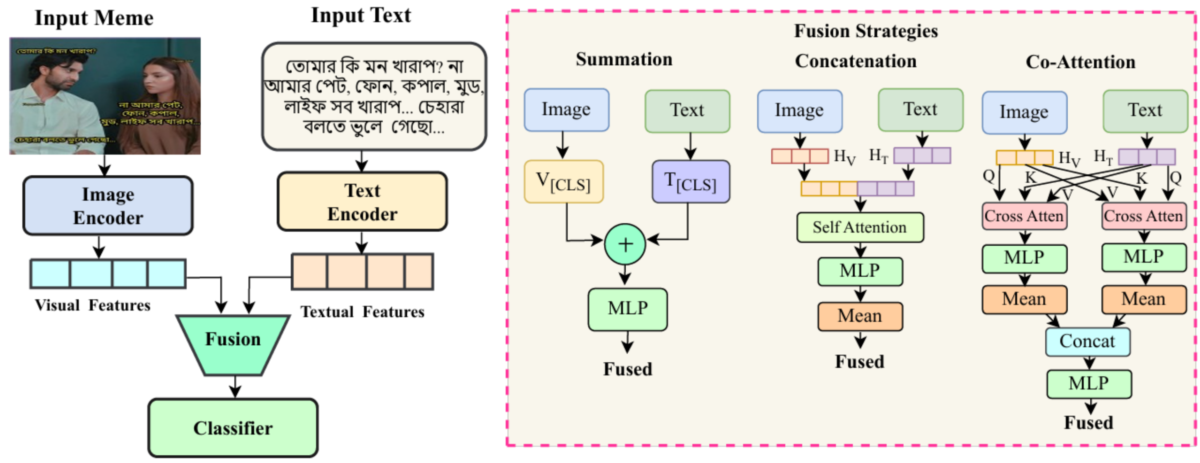


Figure 4: Overview of our hierarchical multimodal framework. Image and text are encoded separately, and their representations are fused using summation, concatenation, or co-attention. The fused features are then used for hierarchical classification across binary labels, hate categories, and targeted groups.

Label Distribution. The BANHATEME dataset contains 2,050 Non-Hate memes and 1,769 Hate memes. Among the hateful memes, Figure 3(a) presents the distribution across hate categories, where Abusive content is most frequent, followed by Political and Gender-based memes, while Religious and Personal Offence are comparatively less common. Figure 3(b) illustrates the breakdown of targeted groups, showing that Communities are the most frequent targets, followed by Individuals and Organizations, with Society-level hate being the least represented.

5 Methodology

Our approach leverages the multimodal nature of memes, which combine visual and textual information to convey meaning. Each meme in our dataset is treated as a multimodal input $x = (x_V, x_T)$, where x_V represents the image content and x_T denotes the extracted text from the meme.

As illustrated in Figure 4, we process each modality through dedicated encoders to obtain modality-specific representations. These representations are then fused via a fusion module to produce a combined multimodal embedding, which is passed into a classification module for prediction. Our implementation utilizes a hierarchical classification loss to improve performance.

5.1 Modality-Specific Representation

To extract meaningful features, we use pretrained encoders tailored to each modality. The image input x_V is processed by a transformer-based vision encoder ϕ_V, which splits the image into patches and appends a special `[CLS]` token representing the entire image. The output is:

$$H_V = \{v_{[\text{CLS}]}, v_1, \ldots, v_m\} = \phi_V(x_V)$$

Similarly, the extracted text x_T is fed into a pretrained text encoder ϕ_T, producing token embeddings including the `[CLS]` token:

$$H_T = \{t_{[\text{CLS}]}, t_1, \ldots, t_n\} = \phi_T(x_T)$$

5.2 Modality Fusion

The representations from both modalities are combined using one of three fusion strategies:

Summation-Based Fusion We sum the `[CLS]` embeddings from both modalities and pass the result through an MLP:

$$h_{\text{fused}} = \text{MLP}(v_{[\text{CLS}]} + t_{[\text{CLS}]})$$

Concatenation-Based Fusion The token embeddings from both modalities are concatenated and processed by a self-attention block followed by an MLP. The fused sequence is mean-pooled to obtain a fixed-length vector:

$$\begin{aligned} H_{\text{fused}} &= \text{MLP}\big(\text{Self-Attention}([H_V; H_T])\big) \\ h_{\text{fused}} &= \text{Mean-Pooling}(H_{\text{fused}}) \end{aligned}$$

Co-Attention Based Fusion Co-attention computes inter-modal attention by using queries from one modality and keys/values from the other. This yields two fused outputs:

Fusion Method	Non-Hate				Hate				Overall	
	P	R	F1	Acc	P	R	F1	Acc	F1	Acc
BanglaBERT + ViT										
Sum based	69.58	81.27	74.97	81.27	72.98	58.76	65.10	58.76	70.40	70.85
Concatenation	69.94	79.81	74.55	79.81	71.96	60.17	65.54	60.17	70.38	70.72
Co-Attention	69.27	66.91	68.07	66.91	63.04	65.54	64.27	65.54	66.31	66.27
BanglaBERT + Swin										
Sum based	72.25	82.96	77.24	82.97	76.11	62.99	68.93	62.99	73.39	73.73
Concatenation	72.31	**85.15**	**78.21**	**85.16**	**78.29**	62.15	**69.29**	62.15	**74.08**	**74.51**
Co-Attention	69.48	74.21	71.76	74.21	67.48	62.15	64.71	62.15	68.50	68.63
XLM-RoBERTa + ViT										
Sum based	62.03	51.68	56.39	51.69	63.62	**72.75**	67.88	**72.75**	62.56	63.01
Concatenation	65.04	64.71	64.88	64.72	59.27	59.60	59.44	59.60	62.36	62.35
Co-Attention	65.71	61.06	63.30	61.07	58.22	62.99	60.52	62.99	62.01	61.96
XLM-RoBERTa + Swin										
Sum based	69.12	82.22	75.11	82.24	73.55	57.33	64.44	57.34	70.18	70.72
Concatenation	**72.75**	76.63	74.64	76.64	71.08	66.66	68.80	66.67	71.94	72.03
Co-Attention	65.05	81.50	72.35	81.51	69.60	49.14	57.62	49.15	65.53	66.54

Table 4: Model benchmarking results on the test split of the BanHateMe dataset are reported. Here, P, R, F1, and Acc represent Precision, Recall, F1 Score, and Accuracy, respectively.

$$H_V^{\text{fused}} = \text{MLP}\left(\sigma\left(\frac{(W_Q H_V)(W_K H_T)^\top}{\sqrt{d_k}}\right)(W_V H_T)\right)$$
$$H_T^{\text{fused}} = \text{MLP}\left(\sigma\left(\frac{(W_Q H_T)(W_K H_V)^\top}{\sqrt{d_k}}\right)(W_V H_V)\right)$$

where W_Q, W_K, W_V are learned projection matrices, σ is the softmax function, and d_k is a scaling factor.

Mean-pooling is applied to each fused representation, which are then concatenated and passed through an MLP to get the final fused vector:

$$h_V^{\text{fused}} = \text{Mean-Pooling}(H_V^{\text{fused}})$$
$$h_T^{\text{fused}} = \text{Mean-Pooling}(H_T^{\text{fused}})$$
$$h_{\text{fused}} = \text{MLP}\left([h_V^{\text{fused}}; h_T^{\text{fused}}]\right)$$

This approach captures detailed interactions between the image and text modalities.

5.3 Classification Module

The fused representation h_{fused}, is passed into a classification head to generate the prediction logits. This classifier is implemented as a simple linear transformation, where the logits are computed using the following operation:

$$\text{logits} = W_{\text{C}} \cdot h_{\text{fused}}$$

5.4 Hierarchical Loss

To train the model in accordance with the hierarchical structure of the classification task, we design a composite loss function. At the core of this formulation is a binary cross-entropy loss, $\mathcal{L}_{\text{binary}}$, which measures the performance of the model in distinguishing between hateful and non-hateful memes.

Once a meme is predicted as hateful, two additional cross-entropy losses are computed. The first, denoted as $\mathcal{L}_{\text{hate\_cat}}$, corresponds to the hate category classification. The second, $\mathcal{L}_{\text{target\_grp}}$, evaluates the model's ability to correctly identify the target group affected by the hate content. The total loss function combines these three components in a weighted manner:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{binary}} + \alpha \cdot \mathcal{L}_{\text{hate\_cat}} + \beta \cdot \mathcal{L}_{\text{target\_grp}}$$

In this equation, α and β are hyperparameters that control the contribution of the hate category and target group classification losses, respectively.

5.5 Experiment Setup

We conducted all experiments on the Kaggle platform using an NVIDIA Tesla P100 GPU with 16 GB VRAM, 32 GB RAM, and 8 CPU cores. For text encoding, we employed two pretrained language models, BanglaBERT (Bhattacharjee et al., 2021) and RoBERTa (Conneau et al., 2019), while for image encoding we used ViT (Dosovitskiy et al., 2020) and Swin Transformer (Liu et al., 2021). We selected these text and vision encoders as they have demonstrated strong performance in Bangla NLP

Fusion Method	Hate Category						Target Group				
	Ab	Po	Ge	Per	Re	Avg	Co	Ind	Org	So	Avg
BanglaBERT + ViT											
Sum based	60.87	64.29	31.03	25.40	32.88	42.89	44.86	58.23	59.20	09.23	42.88
Concatenation	65.74	71.53	52.57	15.79	61.54	53.43	61.32	57.76	63.24	08.13	47.61
Co-Attention	67.23	64.75	40.30	24.24	33.85	46.07	66.67	62.01	67.63	**48.78**	61.27
BanglaBERT + Swin											
Sum based	54.02	**77.61**	32.99	31.84	**76.47**	54.59	59.11	58.25	**78.57**	7.14	50.77
Concatenation	**69.53**	75.56	53.89	19.18	74.00	**58.43**	69.63	56.45	74.45	12.32	53.21
Co-Attention	66.67	71.76	56.65	25.87	66.67	57.52	**71.27**	**65.71**	71.11	38.64	61.68
XLM-RoBERTa + ViT											
Sum based	54.27	45.40	43.98	37.62	10.34	38.32	64.67	60.50	07.23	6.22	34.66
Concatenation	65.38	50.00	45.45	**38.71**	25.35	44.98	63.26	63.76	25.00	29.03	45.26
Co-Attention	64.73	45.83	38.34	18.18	34.48	40.31	61.62	57.00	08.92	32.84	40.10
XLM-RoBERTa + Swin											
Sum based	62.46	70.83	50.37	08.21	68.09	51.99	64.29	60.39	70.27	31.33	56.57
Concatenation	68.80	75.36	50.00	22.50	72.00	57.73	66.94	62.01	77.03	45.78	**62.94**
Co-Attention	63.26	70.34	**57.83**	26.51	64.65	56.52	70.80	65.09	71.14	38.36	61.35

Table 5: Performance across hate categories and target groups on the test split of the BANHATEME dataset, reported using F1 score. Here, Ab, Po, Ge, Per, and Re refer to Abusive, Political, Gender, Personal Offence, and Religious categories, while Co, Ind, Org, and So denote Community, Individual, Organization, and Society, respectively.

and multimodal classification tasks. The MLP layers operated on a 768-dimensional representation, and cross-attention modules also yielded aligned multimodal features of 768 dimensions. All models were fine-tuned for up to 10 epochs with early stopping to prevent overfitting, using a batch size of *16* and learning rate of *2e-5*. For hierarchical loss, we applied weighting parameters with $\alpha = 0.2, 0.8, 0.5, 1.0$ and $\beta = 0.8, 0.2, 0.5, 1.0$ across different levels. Our implementation relied on HuggingFace *Transformers 4.45.1* (Wolf et al., 2020) with *PyTorch 2.4.0* as the backend, and we used *NumPy 1.26.4*, *Pandas 2.2.3*, *Matplotlib 3.7.5*, *Seaborn 0.12.2*, and *scikit-learn 1.2.2* for data processing, analysis, and visualization.

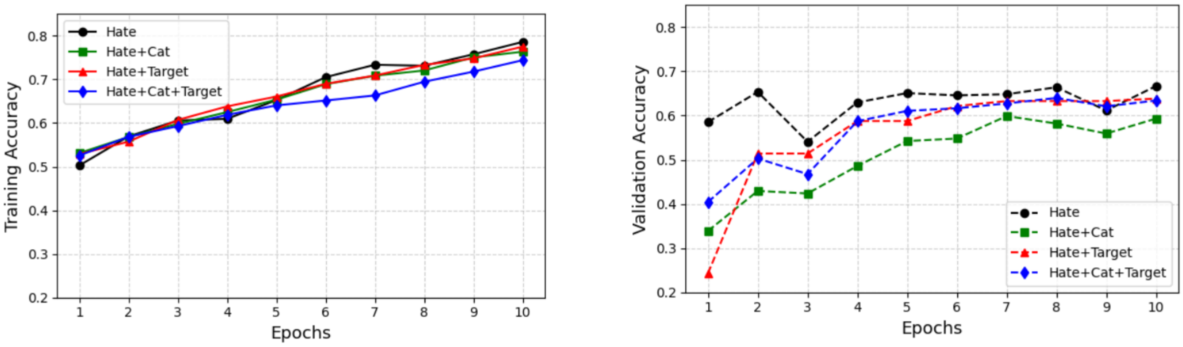
6 Results and Analysis

The performance of different configurations of language and vision models on hate/non-hate detection, hate categories, and targeted groups is reported in Tables 4 and 5. We analyze the results along the following dimensions:

Impact of Language Model. BanglaBERT consistently outperformed XLM-RoBERTa in overall binary detection, achieving gains of about 3-5% in both F1 and accuracy. For hate categories, BanglaBERT showed clear improvements in Political, Religious, and Abusive memes, where performance increased by roughly 4-6%. In contrast, XLM-RoBERTa performed better in Personal Offence and Gender. For targeted groups, BanglaBERT delivered stronger results in Community and Individual prediction, while XLM-RoBERTa achieved higher scores for Organization. Overall, the outcomes indicate that monolingual models are more effective at capturing broad linguistic and cultural cues in Bangla.

Impact of Vision Model. The vision backbone played a crucial role in shaping overall performance. Swin consistently outperformed ViT across most configurations, yielding relative gains of about 2–3% in binary hate detection and up to 6–8% in categories such as Political and Religious. For targeted groups, Swin provided clear advantages in detecting Individuals and Organizations, with improvements ranging from 7–10% over ViT. An exception emerged in Personal Offence, where ViT combined with XLM-RoBERTa achieved a notable score across all settings. Nonetheless, Swin remained the more reliable encoder overall, highlighting the importance of localized visual representations for domains where subtle contextual markers drive hateful interpretation.

Impact of Fusion Strategy. Fusion strategies influenced the performance of the models. For binary detection, concatenation proved most effective, giving BanglaBERT with Swin the highest overall scores and surpassing summation and co-attention by 1–4%. For hate categories, the best method varied: concatenation excelled in Abusive and Personal, summation performed better in Political and Religious, while co-attention achieved the top re-



(a) Training accuracy across categorical cross-entropy loss variants

(b) Validation accuracy across categorical cross-entropy loss variants

Figure 5: Impact of hierarchical loss on model performance, showing smoother convergence and reduced variance compared to single-task and partial supervision settings.

Value of α & β	$\alpha = 0.2, \beta = 0.8$			$\alpha = 0.5, \beta = 0.5$			$\alpha = 0.8, \beta = 0.2$			$\alpha = 1, \beta = 1$		
	H/NH	**Cat**	**Tar**	**H/NH**	**Cat**	**Tar**	**H/NH**	**Cat**	**Tar**	**H/NH**	**Cat**	**Tar**
BanglaBERT + ViT												
Sum based	72.05	51.89	58.75	70.40	42.89	42.88	69.96	56.39	58.75	68.25	49.33	48.06
Concatenation	70.69	53.25	47.84	70.38	53.43	47.61	70.38	53.43	45.58	69.57	59.93	63.42
Co-Attention	67.02	56.39	58.75	66.31	46.07	61.27	67.44	58.68	63.21	65.71	57.01	60.55
BanglaBERT + Swin												
Sum based	73.39	54.59	50.77	73.39	54.59	50.77	70.63	54.26	48.34	66.44	60.57	53.42
Concatenation	70.91	57.10	49.64	74.51	58.43	53.21	72.03	60.40	59.24	66.89	36.77	42.49
Co-Attention	69.78	55.78	61.89	68.50	57.50	61.68	63.97	50.87	49.09	66.99	55.69	63.72
XLM-RoBERTa + ViT												
Sum based	62.73	36.72	31.44	62.56	51.99	56.57	63.53	40.35	31.20	65.65	61.63	65.64
Concatenation	63.84	40.13	37.78	62.36	44.98	45.26	61.29	46.45	36.06	64.16	57.33	54.42
Co-Attention	62.20	36.06	40.48	62.01	40.31	40.10	62.68	36.55	37.42	66.06	49.88	58.11
XLM-RoBERTa + Swin												
Sum based	61.99	52.16	51.59	70.18	51.99	56.57	62.87	53.87	49.87	62.78	23.02	33.23
Concatenation	70.37	57.13	51.54	71.94	57.73	62.94	72.77	63.58	56.04	66.24	57.08	51.74
Co-Attention	63.79	51.02	58.97	65.53	56.52	61.35	64.05	50.23	56.10	64.63	52.93	58.47

Table 6: Impact of α and β in the hierarchical loss. We report the overall performance (F1 score) of detecting hate/non-hate, along with hate category and target group prediction. Here H/NH, Cat, and Tar refer to Hate/Non-Hate, Hate Category, and Target Group prediction results, respectively.

sult in Gender. For targeted groups, co-attention consistently led for Community, Individual, and Society, whereas summation was best for Organization.

Impact of Hierarchical Loss. Figures 5(a) and 5(b) compare models trained on the binary Hate vs. Non-Hate task under different loss formulations. Using only categorical cross-entropy on the binary task yields the highest training accuracy but quickly saturates on validation. Incorporating auxiliary supervision from categories (Hate+Cat) or targets (Hate+Target) slightly reduces training accuracy but stabilizes validation curves. The full hierarchical loss (Hate+Cat+Target) achieves the most consistent validation accuracy with reduced variance across epochs.

Impact of α and β. As shown in Table 6, varying the values of α and β in the hierarchical loss influences the trade-off between hate category and target group prediction. While higher weight on one component improves that sub-task, it typically reduces the other. We find that setting $\alpha = 0.5$ and $\beta = 0.5$ provides the most effective overall performance in terms of F1 score.

Error Analysis. We conduct both quantitative and qualitative error analyses to better understand model behavior across binary, category, and target group levels. A detailed analysis is provided in Appendix C.

7 Conclusion

We present BanHateMe , Bangla hateful meme dataset with hierarchical annotations covering binary labels, hate categories, and targeted groups.

This resource advances multimodal hate detection in low-resource settings by providing culturally grounded annotations and a hierarchical loss that balances predictions across levels with different fusion techniques and their impact on the modality alignment. We envision BANHATEME as a foundation for building culturally aware multimodal moderation systems in Bangla and as a catalyst for future research on hierarchical modeling, fusion strategies, and cross-modal reasoning in underrepresented languages.

Limitations

A primary limitation of our study is the relatively modest dataset size, constrained by the effort required for high-quality hierarchical annotations across categories and targeted groups. While this scale provides a strong foundation, larger datasets would be necessary to further improve generalizability. Another limitation lies in the reliance on pretrained language and vision encoders not originally optimized for Bangla multimodal content. As a result, current models struggle with subtle cultural cues and fine-grained cross-modal interactions. In future work, we plan to expand the dataset through more efficient annotation strategies and explore culturally adapted multimodal architectures tailored to Bangla content. Previous studies on Bangla and multilingual languages (Haider et al., 2024; Fahim et al., 2024; Ahmed et al., 2024) have observed performance variations in large language models (LLMs) across different prompting techniques. In future, we also plan to experiment LVLMs using different techniques to see the impact of the prompt on the performance of LVLMs in our dataset.

Ethical Statement

We collected memes exclusively from publicly accessible social media sources and excluded any content containing explicit nudity or personally identifiable information (PII). All memes were manually reviewed to remove duplicates, unreadable content, or irrelevant material. Annotators were Bangla-speaking individuals familiar with online discourse, and their privacy was strictly maintained; no personal data about them was collected or shared. They were fairly compensated for their work at rates consistent with local norms. To mitigate potential biases, we developed comprehensive annotation guidelines, employed a multi-stage review process, and utilized majority voting to resolve disagreements. Nevertheless, we acknowledge that subjective judgments in hate classification may introduce residual biases. Our dataset is intended solely for research on multimodal hate detection in low-resource languages and should not be misused for malicious purposes. All resources will be released publicly to foster transparency, reproducibility, and future research. We emphasize that any harmful stereotypes or biases in the dataset are unintentional, and we have no intent to harm any individual or community.

References

Fahim Ahmed, Md Fahim, Md Ashraful Amin, Amin Ahsan Ali, and AKM Rahman. 2024. Improving the performance of transformer-based models over classical baselines in multiple transliterated languages. In *ECAI 2024*, pages 4043–4050. IOS Press.

Shawly Ahsan, Eftekhar Hossain, Omar Sharif, Avishek Das, Mohammed Moshiul Hoque, and M Dewan. 2024. A multimodal framework to detect target aware aggression in memes. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2487–2500.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

Rui Cao, Roy Ka-Wei Lee, Wen-Haw Chong, and Jing Jiang. 2023. Prompting for multimodal hateful meme classification. *arXiv preprint arXiv:2302.04156*.

Mohit Chandra, Dheeraj Pailla, Himanshu Bhatia, Aadilmehdi Sanchawala, Manish Gupta, Manish Shrivastava, and Ponnurangam Kumaraguru. 2021. "subverting the jewtocracy": Online antisemitism detection using multimodal deep learning. In *Proceedings of the 13th ACM Web Science Conference 2021*, pages 148–157.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Mithun Das and Animesh Mukherjee. 2023. Banglaabusememe: A dataset for bengali abusive meme classification. *arXiv preprint arXiv:2310.11748*.

Riddhiman Swanan Debnath, Nahian Beente Firuj, Abdul Wadud Shakib, Sadia Sultana, and Md Saiful Islam. 2025. Exmute: A context-enriched multimodal dataset for hateful memes. In *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*, pages 83–89.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Md Fahim, Fariha Tanjim Shifat, Fabiha Haider, Deeparghya Dutta Barua, MD Sakib Ul Rahman Sourove, Md Farhan Ishmam, and Md Farhad Alam Bhuiyan. 2024. Banglatlit: A benchmark dataset for back-transliteration of romanized bangla. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14656–14672.

Raul Gomez, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas. 2020. Exploring hate speech detection in multimodal publications. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1470–1478.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, and Md Farhad Alam. 2024. Banth: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

Eftekhar Hossain, Omar Sharif, and Mohammed Moshiul Hoque. 2022. Mute: A multimodal dataset for detecting hateful memes. 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: student research workshop*, pages 32–39.

Eftekhar Hossain, Omar Sharif, Mohammed Moshiul Hoque, and Sarah M Preum. 2024a. Align before attend: Aligning visual and textual features for multimodal hateful content detection. *arXiv preprint arXiv:2402.09738*.

Eftekhar Hossain, Omar Sharif, Mohammed Moshiul Hoque, and Sarah M Preum. 2024b. Deciphering hate: identifying hateful memes and their targets. *arXiv preprint arXiv:2403.10829*.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. 2020. The hateful memes challenge: Detecting hate speech in multimodal memes. *Advances in neural information processing systems*, 33:2611–2624.

Gitanjali Kumari, Dibyanayan Bandyopadhyay, and Asif Ekbal. 2023. Emoffmeme: identifying offensive memes by leveraging underlying emotions. *Multimedia Tools and Applications*, 82(29):45061–45096.

Fenglin Liu, Yuanxin Liu, Xuancheng Ren, Xiaodong He, and Xu Sun. 2019. Aligning visual regions and textual concepts for semantic-grounded image representations. *Advances in Neural Information Processing Systems*, 32.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Md Ayon Mia, Akm Moshiur Rahman Mazumder, Khadiza Sultana Sayma, Md Fahim, Md Tahmid Hasan Fuad, Muhammad Ibrahim Khan, and Akmmahbubur Rahman. 2025. Banmime: Misogyny detection with metaphor explanation on bangla memes. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17824–17850.

Konstantinos Perifanos and Dionysis Goutsos. 2021. Multimodal hate speech detection in greek social media. *Multimodal Technologies and Interaction*, 5(7):34.

Shraman Pramanick, Dimitar Dimitrov, Rituparna Mukherjee, Shivam Sharma, Md Shad Akhtar, Preslav Nakov, and Tanmoy Chakraborty. 2021. Detecting harmful memes and their targets. *arXiv preprint arXiv:2110.00413*.

Kshitij Rajput, Raghav Kapoor, Kaushal Rai, and Preeti Kaur. 2022. Hate me not: detecting hate inducing memes in code switched languages. *arXiv preprint arXiv:2204.11356*.

Shardul Suryawanshi, Bharathi Raja Chakravarthi, Mihael Arcan, and Paul Buitelaar. 2020. Multimodal meme dataset (multioff) for identifying offensive content in image and text. In *Proceedings of the second workshop on trolling, aggression and cyberbullying*, pages 32–41.

Prashanth Vijayaraghavan, Hugo Larochelle, and Deb Roy. 2021. Interpretable multi-modal hate speech detection. *arXiv preprint arXiv:2103.01616*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Savvas Zannettou, Tristan Caulfield, Jeremy Blackburn, Emiliano De Cristofaro, Michael Sirivianos, Gianluca Stringhini, and Guillermo Suarez-Tangil. 2018. On the origins of memes by means of fringe web communities. In *Proceedings of the internet measurement conference 2018*, pages 188–202.

Yan Zeng, Xinsong Zhang, and Hang Li. 2021. Multi-grained vision language pre-training: Aligning texts with visual concepts. *arXiv preprint arXiv:2111.08276*.

A Annotation Guidelines

This section outlines the guidelines developed for annotating the BANHATEME dataset. Annotators examined both visual and textual elements of each meme to determine whether it should be labeled as Hate or Non-Hate. If annotated as Hate, the meme was further classified into one hate category (Abusive, Political, Gender, Personal offence, Religious) and one targeted group type (Community, Individual, Organization, Society). Figure 6 illustrates representative examples of Hate and Non-Hate memes, along with their assigned categories and target groups.

A.1 General Instructions

The following instructions guided the annotation process:

- **Multimodal Analysis:** Annotators jointly analyzed text and image, considering semantic interplay and cultural context.

- **Hierarchical Labeling:** Each meme was first classified as Hate or Non-Hate. Hate memes were further assigned exactly one hate category and one targeted group type.

- **Annotation Consistency:** Definitions were applied uniformly across the dataset to ensure reliability and reproducibility.

- **Cultural Relevance:** Bangla-specific expressions, code-mixed text, and culturally grounded memes were carefully interpreted.

A.2 Categories and Definitions

Our taxonomy captures five categories of hateful content commonly expressed in Bangla memes. Each category is defined below, along with representative indicators to assist annotation.

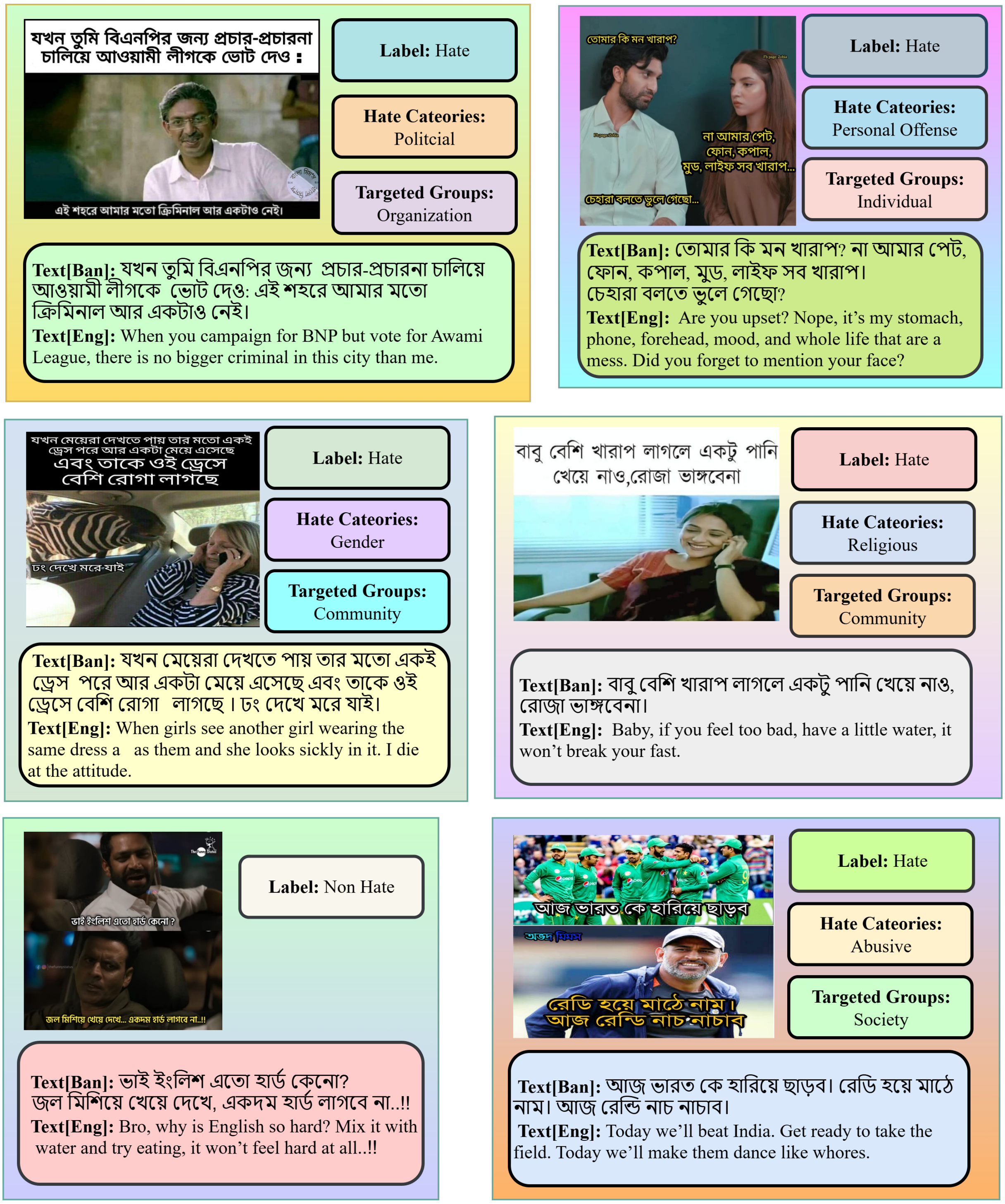
- **Abusive**
  Definition: Content that employs profane, offensive, or degrading language intended to insult, belittle, or provoke hostility, without making explicit threats of physical harm
  Indicators:
  - Use of curse words, slurs, or offensive profanity aimed at individuals or groups
  - Persistent use of hostile, degrading, or insulting language
  - Ridicule or mockery expressed in an aggressive manner, designed to provoke anger, humiliation, or distress

- **Political**
  Definition: Content that targets people or groups based on their political views or affiliations, often through incitement, hostility, or derogatory framing.
  Indicators:
  - Use of dehumanizing metaphors to describe political opponents
  - Explicit calls for violence or exclusion against political group
  - Spread of hostile disinformation to provoke hate or polarization

- **Gender**
  Definition: Content that demeans, stereotypes, or excludes individuals on the basis of gender or gender identity. This includes misogynistic, misandrist, or transphobic expressions that normalize discrimination or gender-based violence.
  Indicators:
  - Use of sexist or patriarchal stereotypes in jokes or insults
  - Justification, encouragement, or trivialization of gender-based violence
  - Derogatory remarks aimed at women, men, or gender-diverse identities

- **Personal Offence**
  Definition: Content that delivers targeted insults or demeaning remarks toward a specific individual, often exploiting personal vulnerabilities, traits, or experiences to humiliate or attack.
  Indicators:
  - Use of derogatory nicknames, personal slurs, or demeaning epithets
  - Mockery of an individual's tragedy, disability, or personal hardship
  - Attacks ridiculing someone's physical appearance or lifestyle

- **Religious**
  Definition: Content that marginalizes or demonizes individuals or communities on the


যখন তুমি বিএনপির জন্য প্রচার-প্রচারনা চালিয়ে আওয়ামী লীগকে ভোট দেও :
এই শহরে আমার মতো ক্রিমিনাল আর একটাও নেই।
তোমার কি মন খারাপ?
না আমার পেট, ফোন, কপাল, মুড, লাইফ সব খারাপ...
চেহারা বলতে ভুলে গেছো...
যখন মেয়েরা দেখতে পায় তার মতো একই ড্রেস পরে আর একটা মেয়ে এসেছে এবং তাকে ওই ড্রেসে বেশি রোগা লাগছে
ঢং দেখে মরে যাই
বাবু বেশি খারাপ লাগলে একটু পানি খেয়ে নাও,রোজা ভাঙ্গবেনা
ভাই ইংলিশ এতো হার্ড কেনো ?
জল মিশিয়ে খেয়ে দেখে... একদম হার্ড লাগবে না..!!
আজ ভারত কে হারিয়ে ছাড়ব
অভদ্র মিমস
রেডি হয়ে মাঠে নাম।
আজ রেন্ডি নাচ নাচাব

Label: Hate

Hate Cateories: Politcial

Targeted Groups: Organization

Text[Ban]: যখন তুমি বিএনপির জন্য প্রচার-প্রচারনা চালিয়ে আওয়ামী লীগকে ভোট দেও: এই শহরে আমার মতো ক্রিমিনাল আর একটাও নেই।

Text[Eng]: When you campaign for BNP but vote for Awami League, there is no bigger criminal in this city than me.

Label: Hate

Hate Cateories: Personal Offense

Targeted Groups: Individual

Text[Ban]: তোমার কি মন খারাপ? না আমার পেট, ফোন, কপাল, মুড, লাইফ সব খারাপ। চেহারা বলতে ভুলে গেছো?

Text[Eng]: Are you upset? Nope, it's my stomach, phone, forehead, mood, and whole life that are a mess. Did you forget to mention your face?

Label: Hate

Hate Cateories: Gender

Targeted Groups: Community

Text[Ban]: যখন মেয়েরা দেখতে পায় তার মতো একই ড্রেস পরে আর একটা মেয়ে এসেছে এবং তাকে ওই ড্রেসে বেশি রোগা লাগছে। ঢং দেখে মরে যাই।

Text[Eng]: When girls see another girl wearing the same dress a as them and she looks sickly in it. I die at the attitude.

Label: Hate

Hate Cateories: Religious

Targeted Groups: Community

Text[Ban]: বাবু বেশি খারাপ লাগলে একটু পানি খেয়ে নাও, রোজা ভাঙ্গবেনা।

Text[Eng]: Baby, if you feel too bad, have a little water, it won't break your fast.

Label: Non Hate

Text[Ban]: ভাই ইংলিশ এতো হার্ড কেনো? জল মিশিয়ে খেয়ে দেখে, একদম হার্ড লাগবে না..!!

Text[Eng]: Bro, why is English so hard? Mix it with water and try eating, it won't feel hard at all..!!

Label: Hate

Hate Cateories: Abusive

Targeted Groups: Society

Text[Ban]: আজ ভারত কে হারিয়ে ছাড়ব। রেডি হয়ে মাঠে নাম। আজ রেন্ডি নাচ নাচাব।

Text[Eng]: Today we'll beat India. Get ready to take the field. Today we'll make them dance like whores.

Figure 6: Examples of annotated memes from the BANHATEME dataset, covering Non-Hate and hateful memes across the Abusive, Political, Gender, Personal Offense, and Religious categories, with their corresponding targeted group annotations.

basis of religious belief, practice, or disbelief. Such content often frames religion as inferior or dangerous to justify exclusion.
Indicators:

- Mockery of religious practices, figures, or rituals
- Advocacy of discrimination or exclusion of religious minorities
- Depicting a religion or its followers as violent, corrupt, or inferior

A.3 Targeted Groups and Definitions

- **Individual:** Hate directed at a specific person, often exploiting characteristics such as gender, popularity, race, or social standing. Targets may include both public figures and private individuals.
- **Organization:** Hate targeting an established body or institution composed of multiple people working toward shared goals. This includes corporations, educational or governmental institutions, and political parties.
- **Community:** Hate directed toward a collective of individuals who share common beliefs, practices, or affiliations. Such groups may be defined by religion, cultural tradition, fandom, or political alignment.
- **Society:** Hate generalized toward a broad population defined by nationality, ethnicity, or geography. This category captures content that vilifies entire societies or nations rather than a single group or organization.

B Annotation Tool

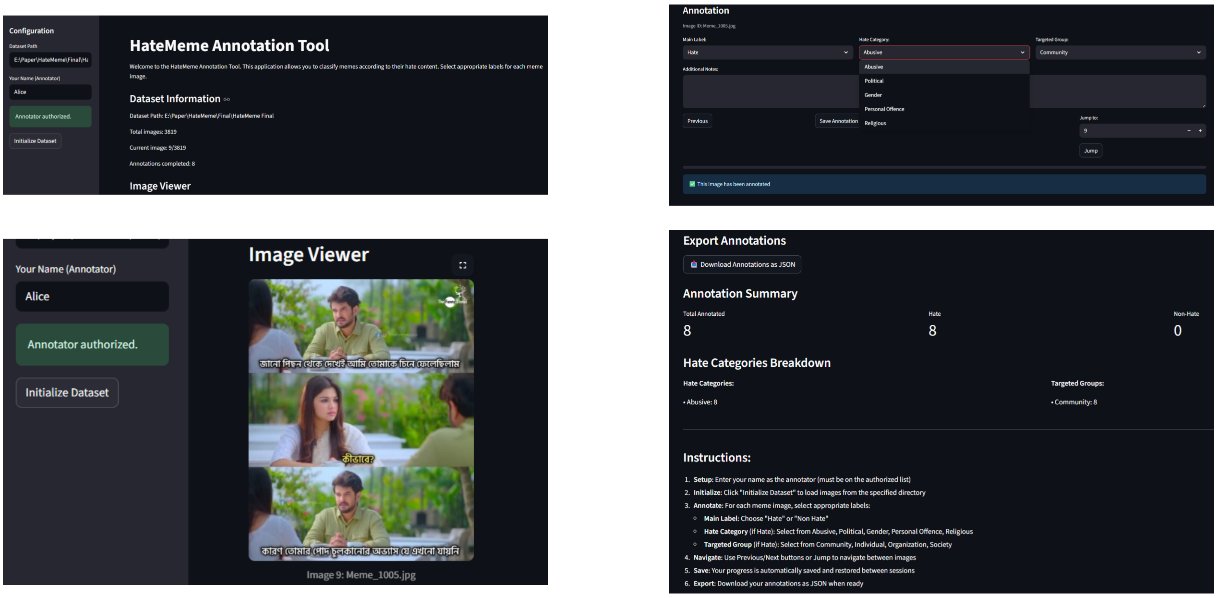
To ensure systematic and reliable labeling, we developed a dedicated web-based annotation tool customized for the BanHateMe dataset. The interface integrates the full hierarchical annotation process and was designed to reduce annotator workload while maintaining consistency across samples. Figure 7 illustrates the user interface, which incorporates several key components to support efficient multi-level annotation.

The platform provides the following core functionalities:

- **Authentication and Setup:** The left sidebar provides secure login, dataset path configuration, and annotation initialization. Access control ensures that only authorized annotators can provide labels.
- **Image Display:** Each meme is shown at its original resolution in the central panel, enabling annotators to examine both visual and textual elements in context.
- **Main Label Selection:** A drop-down menu requires annotators to assign a binary label (Hate vs. Non-Hate). If Non-Hate is chosen, the subsequent menus for hate category and targeted group are automatically set to N/A, preventing mislabeling.
- **Hierarchical Classification (if Hate):**
  - **Hate Category:** A structured drop-down menu enforces single selection from five predefined categories (Abusive, Political, Gender, Personal, or Religious).
  - **Targeted Group:** A second drop-down menu requires annotators to select exactly one of four group types (Community, Individual, Organization, or Society).
- **Contextual Notes:** A free-text field allows annotators to record observations, cultural cues, or rationale for their decisions.
- **Progress Management:** Navigation controls (Previous, Next, Jump) and automatic progress saving facilitate uninterrupted annotation and accurate tracking.
- **Data Export:** Annotations can be exported directly in JSON format, ensuring compatibility with validation scripts and downstream machine learning workflows.

C Error Analysis

Qualitative Analysis on Fusion Strategies. Figure 9 shows how different fusion strategies perform under the BanglaBERT+Swin configuration. Concatenation emerged as the most consistent, particularly in cases where religious or political hate was directed toward communities; by preserving modality-specific signals, it successfully aligned explicit textual references with symbolic visual cues. In contrast, summation often weakened discriminative information, leading to errors in nuanced cases such as personal offence or political hate toward organizations, where subtle textual



Configuration

Dataset Path

E:\Paper\HateMeme\Final\Ha

Your Name (Annotator)

Alice

Annotator authorized.

Initialize Dataset

HateMeme Annotation Tool

Welcome to the HateMeme Annotation Tool. This application allows you to classify memes according to their hate content. Select appropriate labels for each meme image.

Dataset Information

Dataset Path: E:\Paper\HateMeme\Final\HateMeme Final

Total images: 3819

Current image: 9/3819

Annotations completed: 8

Image Viewer

Your Name (Annotator)

Alice

Annotator authorized.

Initialize Dataset

Image Viewer

Image 9: Meme_1005.jpg

Annotation

Image ID: Meme_1005.jpg

Main Label: Hate

Hate Category: Abusive

Targeted Group: Community

Additional Notes:

Jump to: 9

✅ This image has been annotated

Export Annotations

Annotation Summary

Total Annotated	Hate	Non-Hate
8	8	0

Hate Categories Breakdown

Hate Categories:

• Abusive: 8

Targeted Groups:

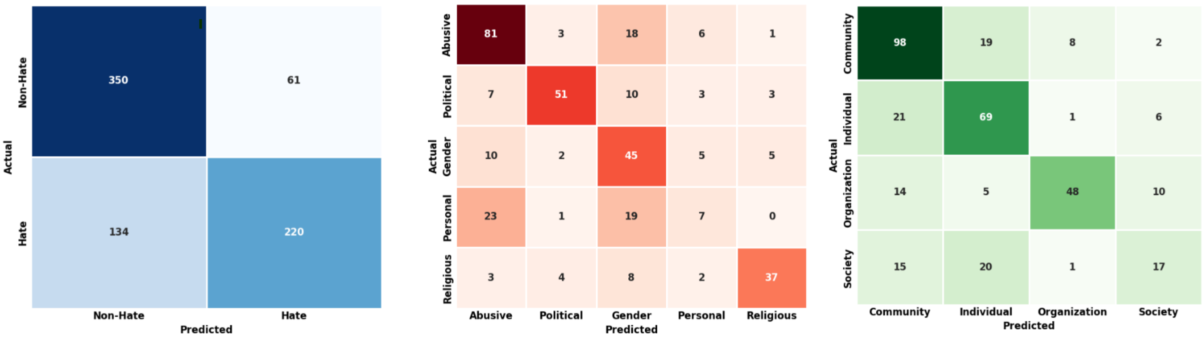
• Community: 8

Instructions:

1. **Setup**: Enter your name as the annotator (must be on the authorized list)
2. **Initialize:** Click "Initialize Dataset" to load images from the specified directory
3. **Annotate**: For each meme image, select appropriate labels:
   - **Main Label:** Choose "Hate" or "Non Hate"
   - **Hate Category** (if Hate): Select from Abusive, Political, Gender, Personal Offence, Religious
   - **Targeted Group** (if Hate): Select from Community, Individual, Organization, Society
4. **Navigate**: Use Previous/Next buttons or Jump to navigate between images
5. **Save**: Your progress is automatically saved and restored between sessions
6. **Export:** Download your annotations as JSON when ready

জানো পিছন থেকে দেখেই আমি তোমাকে চিনে ফেলেছিলাম
কীভাবে?
কারণ তোমার পোদ চুলকানোর অভ্যাস যে এখনো যায়নি
Abusive
Political
Gender
Personal Offence
Religious
Previous
Save Annotation
Jump
Download Annotations as JSON

Figure 7: Interface of the web-based annotation tool for the BanHateMe dataset, highlighting the configuration panel, annotation workspace, and meme display with hierarchical labeling functionality.



(a) Binary Label

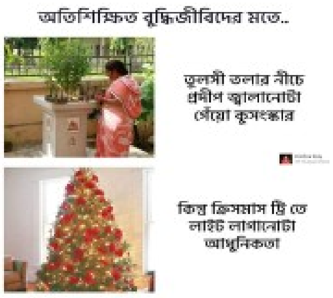
(b) Hate Categories

(c) Targeted Groups

Figure 8: Quantitative error analysis with BanglaBERT + Swin + Concatenation across binary, category, and target group levels.

phrasing or localized visual details were overshadowed by averaged features. Co-attention, while stronger at reasoning over target groups such as society, sometimes over-focuses on a single modality, causing failures like misclassifying offensive gendered caricatures as non-hate. Overall, concatenation proved most reliable for binary and categorical detection, co-attention offered complementary strengths for group-level inference but lacked stability, and summation consistently lagged behind.

Quantitative Error Analysis. We analyze errors for the BanglaBERT + Swin + Concatenation configuration, focusing on binary classification, hate categories, and target groups. In the binary task, as reported in Figure 8(a), nearly 40% of hateful memes are misclassified as non-hate, reflecting the challenge of detecting implicit or sarcastic expressions where cues are subtle. At the category level (Figure 8(b)), Abusive emerges as the most stable class, while Political and Gender often overlap with Abusive, showing leakage of about 15–20%. Personal Offence proves particularly difficult, with more than 30% of instances mislabeled as Abusive or Gender, while Religious hate is sometimes confused with Gender. For target groups (Figure 8(c)), Community is the most consistently recognized, while Individuals show moderate reliability but are redirected into Community about 20% of the time. Organization is less robust, with roughly 25% of its samples misclassified as Community, and Society is the most error-prone, with over 30% of instances mislabeled as either Community or Individual. Overall, these errors indicate that while binary detection is relatively strong, fine-grained categories and target groups remain substantially harder to distinguish due to overlapping linguistic signals and subtle visual markers.



Summation Based
Label: Hate   Categories: Religious   Targeted: Society

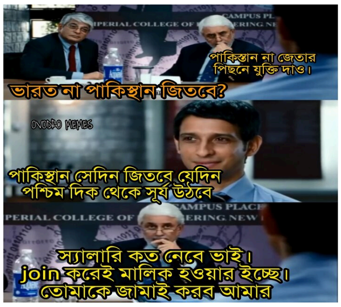
Concatenation Based
Label: Hate   Categories: Religious   Targeted: Community

Co-Attention Based
Label: Hate   Categories: Political   Targeted: Organization

Summation Based
Label: Hate   Categories: Personal Offence Targeted: Individual

Concatenation Based
Label: Hate   Categories: Abusive Targeted: Individual

Co-Attention Based
Label: Hate   Categories: Abusive   Targeted: Society



৯২ ভাতারীদের ভবিষ্যত।😷
MR.ফালতু

Summation Based
Label: Hate   Categories: Gender Targeted: Community

Concatenation Based
Label: Hate Categories: Political Targeted: Community

Co-Attention Based
Label: Non Hate Categories: Gender Targeted: Community

Summation Based
Label: Hate   Categories: Political Targeted: Organization

Concatenation Based
Label: Hate   Categories: Religious   Targeted: Community

Co-Attention Based
Label: Non Hate Categories: Gender Targeted: Community

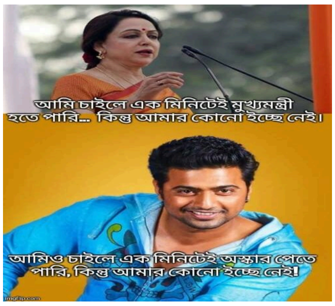

আমি চাইলে এক মিনিটেই মুখ্যমন্ত্রী
হতে পারি... কিন্তু আমার কোনো ইচ্ছে নেই।
আমিও চাইলে এক মিনিটেই অস্কার পেতে
পারি, কিন্তু আমার কোনো ইচ্ছে নেই!
imgflip.com

Figure 9: Qualitative error analysis of fusion strategies under the BanglaBERT+Swin configuration. Green indicates correct predictions, while red indicates errors.

P6Jiggasha: Benchmarking Large Language Models on Bangla Physics Question Answering with Cross-lingual Evaluation

S.M. Shahriar[1], **Md Tahmid Hasan Fuad**[3], **Md Fahim**[2,4,†] **Md. Azad Hossain**[1]

[1]*Chittagong University of Engineering and Technology*
[2]*Center for Computational & Data Sciences, IUB*
[3]*University of Manitoba* [4]*Penta Global Limited*

[†]Team Lead **Correspondence:** {sayeem26s,fahimcse381}@gmail.com

Abstract

Understanding scientific concepts in native languages is crucial for educational accessibility and knowledge transfer. In this work, we present a comprehensive evaluation of Large Language Models (LLMs) on Bangla physics questions, introducing P6Jiggasha, a novel dataset of 1,500 multiple-choice questions compiled from HSC physics textbooks, supplementary guides, admission preparation books, and past examination papers from various educational boards. We evaluate three state-of-the-art models—GPT-4.1, Gemini-2.5 Pro, and DeepSeek-R1-Distill-Llama-70B—on both native Bangla questions and their English translations. Our results reveal significant performance variations, with GPT-4.1 achieving 86.67% accuracy on Bangla questions in a single inference, while other models show substantial improvement through multiple inference attempts, with Gemini-2.5 Pro reaching 89.52% after four iterations. We introduce a *Cumulative Accuracy@k* metric to evaluate iterative reasoning capabilities and provide comprehensive analysis across six physics topics and six question types. Our error analysis reveals systematic cross-lingual inconsistencies where models produce contradictory answers for identical questions across languages. This study provides valuable insights into the capabilities and limitations of current LLMs for low-resource scientific question answering and establishes benchmarks for future research in Bangla natural language processing.

1 Introduction

The democratization of scientific knowledge through native language understanding represents a fundamental challenge in natural language processing and educational technology. While Large Language Models (LLMs) have demonstrated remarkable capabilities across various domains, their performance on scientific content in low-resource languages remains understudied. This gap is particularly pronounced for languages like Bangla, spoken by over 300 million people worldwide, where scientific education primarily occurs in the native language but computational resources remain limited.

Physics education in Bangladesh follows a structured Higher Secondary Certificate (HSC) curriculum, covering concepts from mechanics to quantum physics. Students typically encounter these concepts first in Bangla before transitioning to English-medium higher education, creating unique challenges for both learners and automated systems. Recent advances in LLMs have shown promising results for English scientific question answering (Hendrycks et al., 2020; Lu et al., 2022), but their applicability to non-English scientific content remains largely unexplored (Ahuja et al., 2023; Lai et al., 2023). While multilingual models like GPT-4 (Achiam et al., 2023) and Gemini (Team et al., 2023) present opportunities to bridge this gap, systematic evaluation on domain-specific content in low-resource languages is lacking.

Our contributions are as follows:

- **Dataset Creation and Curation:** We compile P6Jiggasha, the first large-scale Bangla physics MCQ dataset containing 1,500 questions from authentic educational sources with comprehensive English translations and categorizations across six physics domains and six question types.
- **Comprehensive Multi-Model Evaluation:** We systematically evaluate three state-of-the-art LLMs on both Bangla and English versions using the *Cumulative Accuracy@k* metric, analyzing the impact of language modality, question complexity, and iterative inference strategies.
- **Cross-lingual Performance Analysis:** We provide detailed *error analysis* revealing systematic challenges in mathematical reasoning,

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 196–211
December 23, 2025 ©2025 Association for Computational Linguistics

theoretical understanding, and cross-lingual consistency across languages.

Our results reveal nuanced patterns in model performance with significant variations across topics, question types, and languages, providing practical insights for educational technology development and establishing benchmarks for future research in scientific question answering for Bangla and other low-resource languages.

2 Related Work

2.1 Large Language Model Reasoning

The emergence of sophisticated reasoning capabilities in large language models has fundamentally transformed how we approach complex problem-solving tasks. Chain-of-thought prompting demonstrated that generating intermediate reasoning steps significantly improves the ability of large language models to perform complex reasoning (Wei et al., 2022), opening new avenues for scientific reasoning. Recent advances have moved beyond manual prompt engineering, with researchers developing methods for chain-of-thought reasoning without explicit prompting (Wang and Zhou, 2024). The Tree of Thoughts framework (Yao et al., 2023) introduced deliberate decision-making by considering multiple reasoning paths simultaneously, while self-consistency decoding (Wang et al., 2022) improved reasoning reliability by sampling multiple reasoning paths. The development of specialized reasoning models like OpenAI's o1 has further pushed the boundaries, using reinforcement learning to refine problem-solving strategies, with recent surveys highlighting rapid progress in mathematical reasoning capabilities (Zhang et al., 2024).

2.2 Scientific Reasoning Benchmarks

The evaluation of language models on scientific content has become a critical research area, with several landmark datasets establishing the foundation for systematic assessment. Lu et al. introduced multimodal reasoning approaches for science question answering (Lu et al., 2022), while the MMLU benchmark (Hendrycks et al., 2020) provided comprehensive evaluation across multiple domains, including physics. Specialized benchmarks have emerged for mathematical problem-solving (Cobbe et al., 2021) and elementary science questions (Clark et al., 2018). The CMMLU benchmark (Li et al., 2023) extended multilingual evaluation to Chinese, while efforts to develop cross-lingual scientific reasoning benchmarks have highlighted persistent performance gaps in low-resource languages (Bang et al., 2023).

2.3 Bangla Reasoning and NLP Benchmarks

The development of reasoning capabilities for low-resource languages like Bangla has gained momentum through several key initiatives. Early work focused on foundational NLP tasks, with BanglaBERT (Bhattacharjee et al., 2021) establishing baselines for general language understanding. Recent work has specifically targeted reasoning capabilities, with Reveal-Bangla introducing a dataset for cross-lingual multi-step reasoning evaluation (Islam and Sarti, 2025). Additional contributions include SentNoB for sentiment analysis (Islam et al., 2021), BanglaT5 for text generation and translation (Bhattacharjee et al., 2022), BanglaBook for large-scale sentiment analysis (Kabir et al., 2023), and BanglaNLG for natural language generation (Bhattacharjee et al., 2022). The BanglaParaphrase dataset (Akil et al., 2022) has contributed to semantic understanding tasks, while IndicNLPSuite (Kakwani et al., 2020) provides broader South Asian language benchmarks. However, domain-specific reasoning benchmarks in scientific subjects like physics remain largely unexplored, creating the gap our work addresses.

3 P6Jiggasha Dataset

3.1 Data Collection

We systematically collected 1,500 multiple-choice physics questions from diverse Bangladeshi educational sources spanning various academic levels and institutional contexts. Figure 1 illustrates our complete data generation workflow consisting of four main stages: collection and OCR extraction, structure and preprocessing, filtering, and verification with categorization. We systematically collected 1,500 multiple-choice physics questions from diverse Bangladeshi educational sources including HSC Physics Textbooks, Physics Guides, Engineering Question Banks, and Test Papers. Each question follows a standard multiple-choice format with four options (A, B, C, D) and a single correct answer, covering easy-to-difficult MCQs with comprehensive physics concepts. All correct answers were rigorously re-checked directly against the original sources and retained only when the match was unambiguous.

Figure 2 shows the distribution of questions

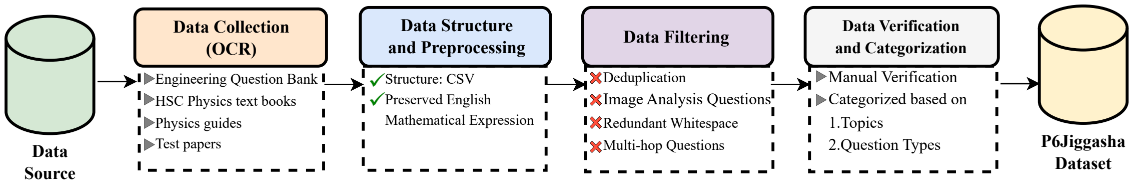


Figure 1: Data generation workflow for the P6Jiggasha dataset showing the pipeline from collection through OCR extraction, filtering, and verification to final categorization across physics topics and question types.

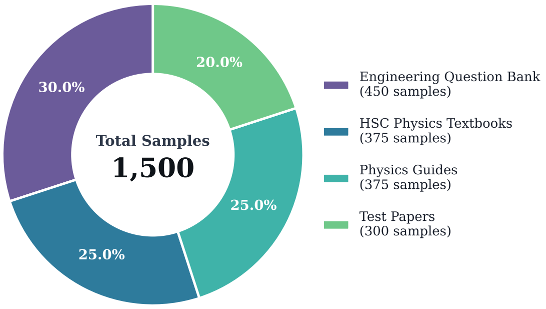


Figure 2: Source distribution showing Engineering Question Banks (30%), HSC Physics Textbooks (25%), Physics Guides (25%), and Test Papers (20%) representing diverse educational materials.

collected from various educational sources across Bangladesh's academic ecosystem, reflecting the diversity of materials used in physics education.

3.2 OCR and Data Extraction

The source materials were primarily available in PDF format and printed documents. We employed advanced language models including Grok 4 and GPT-5 for Optical Character Recognition (OCR), to extract questions from these diverse sources. The OCR process involved automated text extraction from PDF documents, image-to-text conversion for scanned materials, mathematical expression recognition and formatting, and rigorous quality verification of extracted content to ensure accuracy and completeness of physics terminology and mathematical notations.

3.3 Data Structure and Pre-processing

The dataset is structured in CSV format with preserved English mathematical expressions. During the pre-processing phase, we implemented comprehensive data filtering to remove duplications, graph/image-based questions, redundant whitespace, and multi-hop questions. This filtering process ensures consistency and maintains focus on text-based physics concepts while preserving the integrity of mathematical expressions essential for physics problem-solving.

Data Distribution	**Samples**
Total Dataset	1,500
Test Set (allocated from total)	500
Text Statistics	
Mean Word Count	23.2
Maximum Word Count	46
Minimum Word Count	4

Table 1: Dataset composition with 1,500 total questions (500 for testing) and text statistics showing mean word count of 23.2, reflecting concise question formulations.

3.4 Data Verification and Categorization

Following the data filtering phase, we enhanced the dataset through manual verification and systematic categorization. All mathematical expressions and physics terminology were manually verified for accuracy by domain experts. Using Gemini 2.5 Pro, we analyzed each question to assign appropriate topic and question type labels, ensuring comprehensive coverage across different physics domains and cognitive complexity levels.

Table 1 presents the dataset composition and statistical overview, with 500 questions allocated for testing purposes and comprehensive text statistics showing the linguistic characteristics of the dataset.

We categorized the dataset along two primary dimensions based on systematic analysis:

Topics (six categories): Electromagnetism, Mechanics, Thermodynamics, Wave Optics, Quantum Physics, and Modern Physics, reflecting comprehensive coverage of fundamental and advanced physics concepts.

Question Types (six categories): Based on the distribution analysis, we identified and categorized questions into Mathematical, Theoretical, Definition, Reasoning, Application, and Miscellaneous types, capturing different cognitive demands and problem-solving approaches.

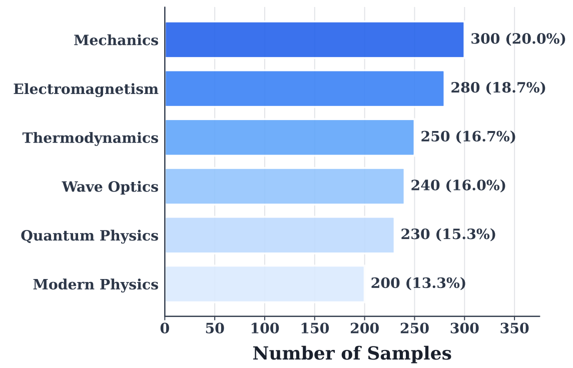


Figure 3: Physics topic distribution with Mechanics dominating (300 samples, 20%) followed by Electromagnetism (280, 18.7%), Thermodynamics (250, 16.7%), Wave Optics (240, 16%), Quantum Physics (230, 15.3%), and Modern Physics (200, 13.3%).

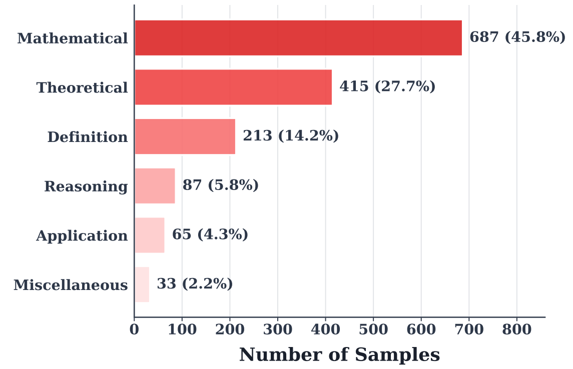


Figure 4: Question type distribution showing Mathematical questions as the largest category (687, 45.8%), followed by Theoretical (415, 27.7%), Definition (213, 14.2%), Reasoning (87, 5.8%), Application (65, 4.3%), and Miscellaneous (33, 2.2%).

Figure 3 presents the distribution across physics topics, showing that Mechanics dominates the topic distribution with 300 samples, followed by Electromagnetism (280 samples) and Thermodynamics (250 samples). Figure 4 shows the question type distribution, where Mathematical questions constitute the largest portion with 687 samples, reflecting the quantitative nature of physics education, while Miscellaneous questions represent the smallest category with 33 samples.

4 Experiment Setup

4.1 Bangla QA based Prompting

We employ *zero-shot prompting* to evaluate the reasoning capabilities of large language models (LLMs). For each question instance, the input to the model consists of three components: a system prompt P, the Bangla natural language question Q_{BAN}, and a set of candidate answer options $O_{\text{BAN}} = \{o_1, o_2, o_3, o_4\}$ where every o_i is given in Bangla. The model is tasked with selecting the optimal answer $o^* \in O_{\text{BAN}}$.

The system prompt P is designed to encourage deliberative reasoning by instructing the model to think step-by-step before committing to a final answer. This aligns with recent trends in prompt engineering where reasoning-based instructions can improve model performance, particularly in multistep or ambiguous queries.

We evaluate three state-of-the-art LLM families in this setup: `GPT`, `Gemini`, and `DeepSeek`. All models are evaluated under identical prompting conditions to ensure fair comparison. The detailed prompt is given in the Appendix.

4.2 Translation-based Prompting

Most reasoning-capable large language models (LLMs) exhibit a strong bias toward English-language reasoning, primarily due to the predominance of English data in their pretraining corpora. Motivated by this, we also evaluate the performance of these models on an English-translated version of our dataset.

To create this version, we translated both the questions and the corresponding answer options into English. The prompt used for translation is provided in the Appendix. For this task, we employ `Gemini-2.5 Pro` and manually verify the generated translations, making corrections where necessary to ensure accuracy and consistency.

Following translation, we conduct zero-shot prompting experiments analogous to our original setup. Each input to the model consists of three components: a system prompt P, an English natural language question Q_{ENG}, and a set of answer options $O_{\text{ENG}} = \{o_1, o_2, o_3, o_4\}$, where each o_i is provided in Bangla. The model is tasked with selecting the optimal answer $o^* \in O_{\text{ENG}}$.

4.3 Evaluation Metrics

We evaluate model performance using a retry-based metric we term **Cumulative Accuracy@k** (Cumul Acc@k), where $k \in \{1, 2, 3, 4\}$. Unlike the standard Pass@k metric, which assumes access to k simultaneous guesses per sample, our approach allows the model up to k sequential attempts to answer each question correctly.

Formally, let S_1 be the original set of N questions. At each round $t \in \{1, 2, \ldots, k\}$, we define

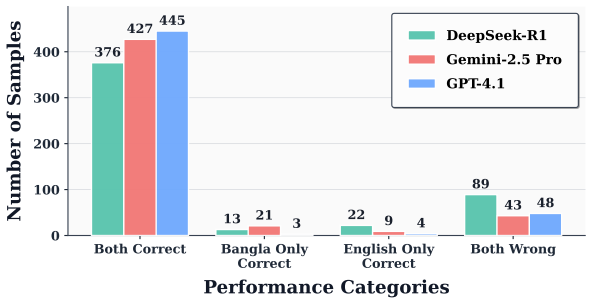


Figure 5: Cross-linguistic performance comparison across four categories: Both Correct, Bangla Only Correct, English Only Correct, and Both Wrong. GPT-4.1 achieves the highest consistency (445 both correct), while DeepSeek-R1 shows the most failures (89 both wrong).

S_t as the subset of questions not answered correctly in any of the previous $t-1$ rounds. In each round, the model is re-prompted with only the remaining incorrect samples from the prior round. Let $a_t(q)$ denote the model's prediction for question q in round t, and let o_q^* be the correct answer.

The Cumulative Accuracy@k is defined as:

$$\text{Cumul Acc@}k = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big\{ \exists t \le k \text{ such that } a_t(q_i) = o_{q_i}^* \Big\}$$

This metric captures the proportion of total questions that the model eventually answers correctly within k retries. It reflects a realistic, interactive usage scenario where a system is allowed to reattempt difficult questions rather than relying solely on a single response. We report Cumulative Accuracy@k for $k = 1, 2, 3, 4$.

5 Results and Analysis

Our evaluation of state-of-the-art LLMs on P6Jiggasha reveals substantial performance gaps and distinct behavioral patterns across models, languages, and question types, with resubmission strategies proving particularly effective for initially lower-performing models.

Cross-Linguistic Consistency Analysis. Figure 5 reveals multilingual robustness patterns across models. GPT-4.1 exhibits the strongest consistency with 445 questions correct in both languages, followed by Gemini-2.5 Pro (427) and DeepSeek-R1 (376). Language-specific patterns show GPT-4.1 with minimal Bangla-only successes (3) versus higher English-only successes (4), while DeepSeek-R1 and Gemini-2.5 Pro show balanced language-specific performance (13-21 and 22-9, respectively). The "Both Wrong" category highlights fundamental gaps, with DeepSeek-R1 (89) showing significantly more failures than GPT-4.1 (48) and Gemini-2.5 Pro (43).

Table 2 presents the comprehensive results across all evaluated models and languages.

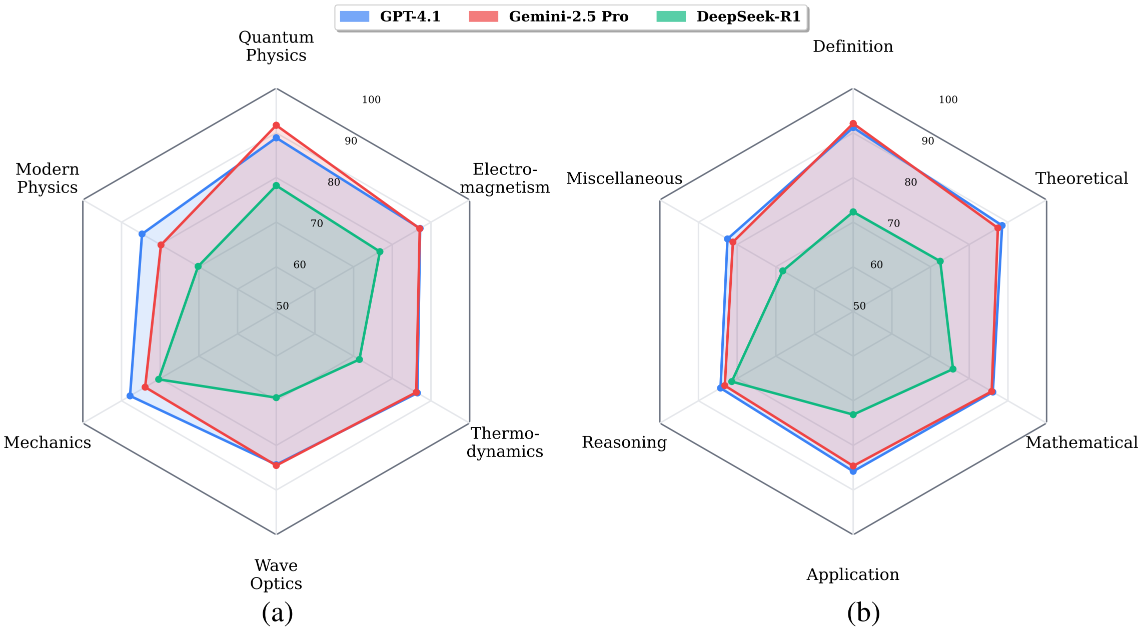
LLMs Comparison. Table 2 reveals significant performance variations across models in the initial inference round (*Cumul Acc@1*). GPT-4.1 demonstrates superior performance on Bangla questions with 86.67% accuracy, substantially outperforming Gemini-2.5 Pro (71.24%) and DeepSeek-R1 (48.00%). This 38.67 percentage point gap between the best and worst performing models highlights the varying capabilities of current LLMs in handling Bangla physics content. Notably, the performance hierarchy (GPT-4.1 $>$ Gemini-2.5 Pro $>$ DeepSeek-R1) remains consistent across both languages, though the magnitude of gaps varies, with DeepSeek-R1 showing particularly weak initial performance in Bangla compared to English.

Impact of Iterative Re-evaluation Strategy. The cumulative accuracy improvements across multiple inference iterations (*Cumul Acc@2–4*) demonstrate the effectiveness of the resubmission strategy. Gemini-2.5 Pro shows the most dramatic improvement, gaining 18.28 percentage points from *Cumul Acc@1* (71.24%) to *Cumul Acc@4* (89.52%), ultimately matching GPT-4.1's final performance. DeepSeek-R1 exhibits the largest absolute gain of 29.71 percentage points in Bangla, though still achieving the lowest final accuracy at 77.71%. GPT-4.1 shows the smallest improvement (2.85 percentage points), indicating high initial accuracy with limited room for enhancement through resubmission. The diminishing returns pattern across iterations (largest gains at *Cumul Acc@2*, smaller at subsequent attempts) suggests that most correctable errors are resolved within the first retry, with marginal benefits from additional resubmissions.

Cross-lingual Performance Analysis. English translation yields mixed results across models. GPT-4.1 shows minimal language dependency with similar performance in both languages (86.67% Bangla *vs.* 84.57% English in *Cumul Acc@1*), ultimately achieving slightly higher final accuracy in English (89.90% *vs.* 89.52%). Conversely, Gemini-2.5 Pro performs better with Bangla questions ini-

Model	Language	Cumulative Accuracy (%)			
		Cumul Acc@1	Cumul Acc@2	Cumul Acc@3	Cumul Acc@4
GPT-4.1	Bangla	86.67	88.57	89.00	89.52
GPT-4.1	English	84.57	88.38	89.14	89.90
Gemini-2.5 Pro	Bangla	71.24	78.86	84.95	89.52
Gemini-2.5 Pro	English	68.32	83.81	85.52	87.25
DeepSeek-R1	Bangla	48.00	72.00	72.19	77.71
DeepSeek-R1	English	62.67	65.52	78.67	79.50

Table 2: Overall model performance across languages and inference iterations, showing GPT-4.1's high initial accuracy with minimal improvement, Gemini-2.5 Pro's dramatic gains through resubmission (+18.28%), and DeepSeek-R1's substantial but insufficient improvement (+29.71% in Bangla).



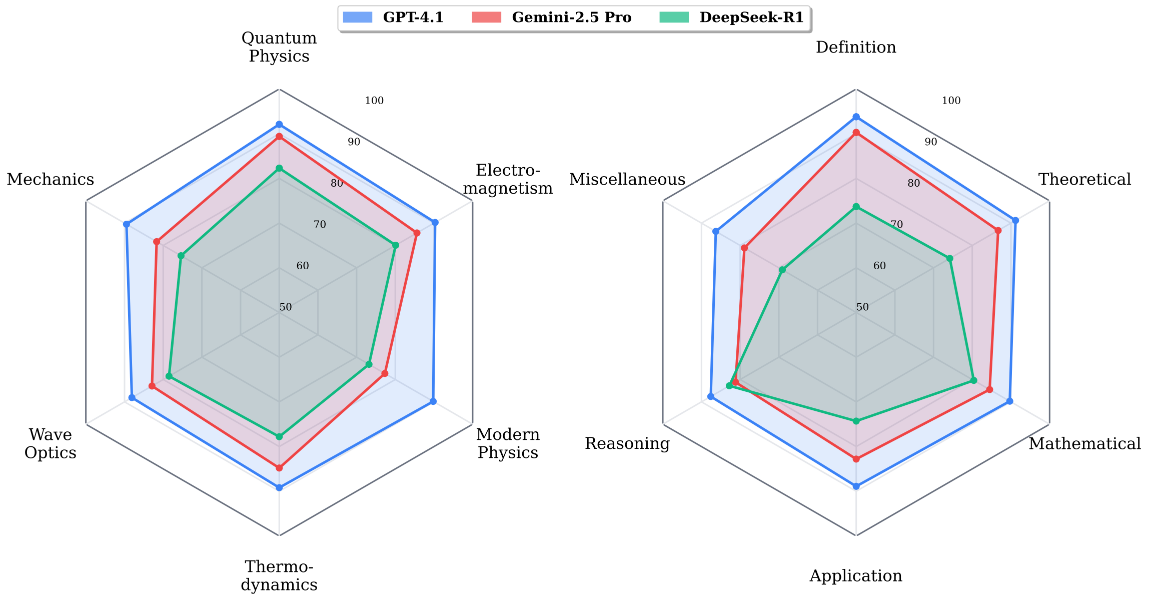
(a)

(b)

Figure 6: Performance analysis for Bangla questions: (a) Topic-wise performance showing consistent strength in Quantum Physics and Electromagnetism versus challenges in Modern Physics and Mechanics, (b) Question type-wise performance revealing strong accuracy on Definition/Theoretical questions but progressive degradation for Application/Reasoning tasks.

tially (71.24% *vs.* 68.32%) but shows stronger improvement trajectories in English during resubmissions, reaching 87.25% final accuracy. DeepSeek-R1 demonstrates a notable preference for English at initial inference (62.67% *vs.* 48.00%, a 14.67 percentage point gap), though Bangla performance catches up through multiple iterations (77.71% *vs.* 79.50% final), suggesting language-specific reasoning patterns and imbalanced training data distribution that can be partially compensated through iterative refinement.

Topic-wise Performance Analysis. Figure 6(a) and Figure 7(a) show topic-wise performance on Bangla and English questions using *Cumul Acc@4*. All models perform better on *Quantum Physics* and *Electromagnetism*, with GPT-4.1 and Gemini-2.5 Pro achieving 88–95% accuracy across most domains. *Modern Physics* and *Mechanics* remain more challenging, especially for DeepSeek-R1, which trails by 10–20 percentage points. DeepSeek-R1 also struggles notably with *Wave Optics* on Bangla questions, showing the lowest accuracy among topics. Overall, radar charts indicate stable performance hierarchies, with GPT-4.1 leading consistently across all six domains. Cross-linguistically, models perform slightly better and more consistently on Bangla, while English results exhibit more compressed performance ranges.



(a) (b)

Figure 7: Performance analysis for English questions: (a) Topic-wise performance showing similar domain trends as Bangla with slightly compressed ranges, (b) Question type-wise performance confirming that cognitive complexity outweighs linguistic factors.

Question Type-wise Performance Analysis. Figure 6(b) and Figure 7(b) present type-wise performance on Bangla and English questions. All models excel on *Definition* questions, with GPT-4.1 and Gemini-2.5 Pro near 90% and DeepSeek-R1 around 85%. For *Mathematical* questions, GPT-4.1 leads at 95%, Gemini-2.5 Pro follows at 90%, and DeepSeek-R1 remains competitive at 82%, showing relative strength on reasoning-based queries. DeepSeek-R1 attains about 78% on *Mathematical* questions for both languages, reflecting moderate math solving ability. However, it performs worst on *Miscellaneous* questions (60%), marking the lowest score across all types and revealing challenges with unconventional formats. Consistent trends across both languages indicate that question-type complexity, not language, primarily drives performance differences.

Design Choice	Cumul@k Acc			Pass@k Acc	
	C@1	**C@2**	**C@3**	**Pass@1**	**Pass@1**
Zero Shot					
GPT-4.1	86.67	88.57	89.00	87.39	89.57
Gemini-2.5 Pro	71.24	78.86	84.95	73.64	76.32
DeepSeek-R1	48.00	72.00	72.19	55.28	67.92
CoT					
GPT-4.1	88.94	92.35	92.87	88.25	90.88
Gemini-2.5 Pro	76.58	84.02	89.72	77.48	82.14
DeepSeek-R1	56.57	78.93	83.83	68.27	76.39

Table 3: Comparison of different prompt in P6JIGGASHA dataset. We consider Bangla Question for this experiment. Here **C@k** means *Cumulative@k Acc*

LLM Prompting Results. Table 3 compares the performance of GPT-4.1, Gemini-2.5 Pro, and DeepSeek-R1 on Bangla questions from the P6JIGGASHA dataset under Zero-Shot and Chain-of-Thought (CoT) prompting. Across all metrics, GPT-4.1 consistently outperforms the other models, achieving high cumulative and pass@k accuracies even in the zero-shot setting, with further improvements under CoT prompting. CoT prompts benefit all models, with smaller models like DeepSeek-R1 showing the largest relative gains, indicating that reasoning prompts help weaker models more significantly. As expected, both cumulative@k and pass@k accuracies increase with k, reflecting the higher probability of obtaining a correct answer when multiple candidates are considered. Overall, the results highlight that larger models perform well even without reasoning prompts, while CoT prompts and multiple candidate outputs further enhance performance across models.

6 Error Analysis

We conducted a comprehensive error analysis examining representative failure cases across question types and physics topics. Detailed breakdowns are provided in Appendix A. Our findings reveal performance gaps as well as deeper structural weaknesses in cross-lingual physics reasoning, especially where linguistic variation interacts with domain-specific concepts.

Mathematical and Theoretical Challenges. Figure 8(a,e) shows universal success for straightforward calculations but reveals Gemini-2.5 Pro's inconsistent wave optics encoding. Figure 8(b,d) demonstrates cross-lingual inversions, indicating language-specific rather than unified formula retrieval (see Appendix A, subsection A.1–A.2). These errors suggest reliance on pattern-matching over conceptual understanding, with theoretical items exposing persistent fragility even under multiple inference attempts. Such weaknesses indicate that deeper symbolic or conceptual grounding remains limited across models.

Applied and Practical Reasoning. Figure 8(c,f) shows models performing well in Bangla but collapsing in English, particularly for thermodynamic comfort and engine efficiency (Appendix A, subsection A.3–A.4). These failures illustrate that real-world physics reasoning is highly sensitive to linguistic framing, where small contextual shifts produce divergent interpretations. This highlights that applied problems require robust world-knowledge alignment, which current LLMs struggle to maintain consistently across languages.

Cross-lingual Inconsistencies. Examples (c–f) in Figure 8 show inconsistent answers across languages for identical problems (Appendix A, subsection A.5). This suggests parallel but weakly aligned internal knowledge representations, where identical concepts trigger different reasoning paths depending solely on query language. Such instability poses concerns for bilingual learners who may switch between Bangla and English during study.

Iterative Reasoning Improvements. Figures 9 and 10 show DeepSeek-R1's +29.71% gain through cumulative accuracy, adding verification layers across attempts (Appendix A, subsection A.6). Although multi-round prompting stabilizes reasoning, improvements vary by domain, revealing that many initial errors stem from incomplete intermediate steps. Iterative refinement helps correct shallow mistakes but cannot reliably resolve deeper conceptual gaps.

Translation-Induced Failures. Figure 11 highlights semantic drift during translation that affects performance independently of physics ability (Appendix B). Even with manual verification, subtle phrasing changes can shift interpretation, adding instability—especially in applied or context-sensitive tasks. This indicates that translation artifacts compound existing reasoning vulnerabilities, disproportionately affecting cross-lingual evaluation.

7 Conclusion

This work presents the first comprehensive evaluation of Large Language Models on Bangla physics questions, introducing the `P6Jiggasha` dataset of 1,500 multiple-choice questions across six physics topics and question types. We evaluate three state-of-the-art models—GPT-4.1, Gemini-2.5 Pro, and DeepSeek-R1-Distill-Llama-70B—revealing significant performance variations across languages and complexity levels. We introduce the *Cumulative Accuracy@k* metric for sequential reasoning evaluation and identify systematic challenges in mathematical reasoning, theoretical understanding, and cross-lingual consistency. Our findings highlight that language modality significantly impacts model performance, emphasizing the need for balanced multilingual training data in scientific domains. Future work should explore open-ended formats, multimodal integration, and expansion to other scientific domains and low-resource languages.

Limitations

Our evaluation focuses exclusively on multiple-choice questions, which may not fully capture scientific reasoning required for open-ended explanations. The dataset currently excludes visual elements such as graphs and diagrams; we plan to incorporate multimodal capabilities in future work. Our study is confined to physics, and generalizability to other STEM subjects in Bangla remains uncertain. The observed cross-lingual inconsistencies raise reliability concerns for educational deployment. Furthermore, the English translations—although manually checked—may still introduce subtle semantic shifts that influence model performance, especially for context-dependent reasoning tasks. Our iterative evaluation strategy (CumulAcc@k) improves robustness but does not fully disentangle genuine reasoning improvements from repeated exposure effects.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Kabir Ahuja, Harshita Diddee, Rishav Hada, Millicent Ochieng, Krithika Ramesh, Prachi Jain, Akshay Nambi, Tanuja Ganu, Sameer Segal, Maxamed Axmed, et al. 2023. Mega: Multilingual evaluation of generative ai. *arXiv preprint arXiv:2303.12528*.

Ajwad Akil, Najrin Sultana, Abhik Bhattacharjee, and Rifat Shahriyar. 2022. Banglaparaphrase: A high-quality bangla paraphrase dataset. *arXiv preprint arXiv:2210.05109*.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, and Rifat Shahriyar. 2022. Banglanlg and banglat5: Benchmarks and resources for evaluating low-resource natural language generation in bangla. *arXiv preprint arXiv:2205.11081*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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.

Khondoker Ittehadul Islam and Gabriele Sarti. 2025. Reveal-bangla: A dataset for cross-lingual multi-step reasoning evaluation. *arXiv preprint arXiv:2508.08933*.

Mohsinul Kabir, Obayed Bin Mahfuz, Syed Rifat Raiyan, Hasan Mahmud, and Md Kamrul Hasan. 2023. Banglabook: A large-scale bangla dataset for sentiment analysis from book reviews. *arXiv preprint arXiv:2305.06595*.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul NC, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020. Indicnlpsuite: 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.

Viet Dac Lai, Nghia Trung Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. 2023. Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning. *arXiv preprint arXiv:2304.05613*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *Advances in Neural Information Processing Systems*, 37:66383–66409.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36:11809–11822.

Janice Ahn Zhang, Aidan Ning, Zhiyuan Jiao, Zijian Ma, Yikang Chen, Jiayi Zhang, et al. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

A Detailed Error Analysis with Examples

This section provides comprehensive analysis of model failure patterns observed in the P6Jiggasha dataset, with detailed examination of representative cases across different physics domains and question types.

A.1 Mathematical Reasoning Challenges

Figure 8 presents representative failure cases across different physics domains. Example (a) demonstrates universal success across all models (B, B, B in both languages) for a *Young's modulus* calculation, indicating strong capability when problems require straightforward formula application with clearly identified variables and direct substitution. Example (e) reveals model-specific challenges in wave optics interference calculations involving intensity superposition.

While GPT-4.1 and DeepSeek-R1 correctly identify option A in both languages, Gemini-2.5 Pro consistently fails, selecting B in Bangla and D in English. This language-specific error pattern suggests Gemini struggles with the conceptual understanding of intensity calculations in interference, incorrectly applying linear summation rather than accounting for phase relationships. The fact that Gemini selects different wrong answers across languages indicates the error mechanism itself is language-dependent, suggesting inconsistent encoding of wave optics principles across multilingual representations.

A.2 Theoretical Concept Understanding

In Figure 8(b), the test examines the relationship between diffraction and interference, where DeepSeek-R1 incorrectly selects refraction (D) in Bangla while correctly identifying interference (C) in English. GPT-4.1 and Gemini-2.5 Pro maintain consistency with correct answers in both languages. This suggests DeepSeek's Bangla physics vocabulary may conflate closely related optical phenomena, indicating weaker terminological precision in low-resource language understanding.

Figure 8(d) reveals dramatic cross-lingual inconsistencies in *moment of inertia* formula retrieval. In Bangla, GPT-4.1 fails (A) while Gemini-2.5 Pro and DeepSeek-R1 succeed (D, D). In English, this pattern completely inverts: GPT-4.1 succeeds (D) while Gemini-2.5 Pro and DeepSeek-R1 both fail (A, A). This systematic cross-lingual inversion—where models that succeed in one language fail in the other—provides strong evidence that physics formulas are not stored in language-agnostic representations but rather encoded separately for each language, leading to inconsistent retrieval depending on query language.

A.3 Applied Reasoning and Contextual Understanding

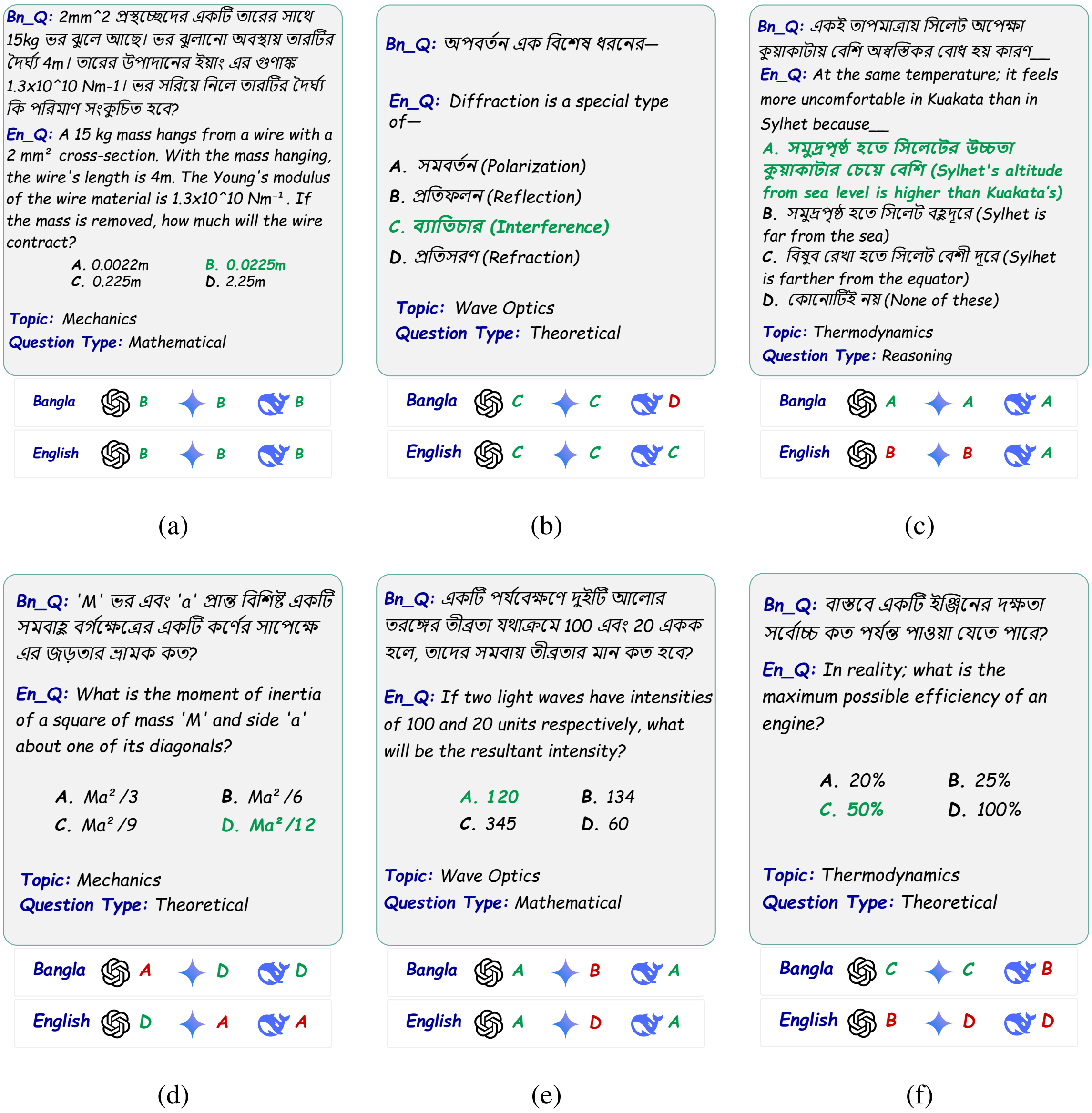
Figure 8(c) demonstrates interesting cross-lingual performance patterns in thermodynamic reasoning about altitude and thermal comfort. All models correctly identify option A in Bangla, but in English, GPT-4.1 and Gemini-2.5 Pro both incorrectly select B (distance from sea), while only DeepSeek-R1 maintains correct reasoning. This suggests that contextual real-world physics applications may be better encoded in Bangla training data for GPT and Gemini, or that English translation introduces semantic ambiguity that misleads these models. The fact that DeepSeek maintains consistency while larger models fail in English is particularly noteworthy, suggesting different architectural or training approaches to handling context-dependent reasoning.

A.4 Practical Physics Understanding

Figure 8(f) examines understanding of realistic engine efficiency limits, revealing significant challenges across all models. In Bangla, GPT-4.1 and Gemini-2.5 Pro both correctly identify C (50%) while DeepSeek-R1 fails (B, 25%). However, in English, *all three models fail*, with GPT-4.1 selecting B (25%), and both Gemini-2.5 Pro and DeepSeek-R1 selecting D (100%). This complete performance collapse in English—where even previously successful models fail—suggests that the conceptual bridge between idealized thermodynamic principles (Carnot efficiency) and practical engineering constraints is poorly represented in English training data. The scattered predictions across different percentage values indicate models confuse theoretical maximums, practical limits, and unrealistic ideal cases, highlighting systematic gaps in applied thermodynamics reasoning.

A.5 Cross-lingual Consistency Issues

Four examples in Figure 8—(c), (d), (e), and (f)—exhibit significant prediction inconsistencies between Bangla and English versions. Example (d) shows the most striking pattern: a complete performance inversion where successful models in



Bn_Q: 2mm^2 প্রস্থচ্ছেদের একটি তারের সাথে 15kg ভর ঝুলে আছে। ভর ঝুলানো অবস্থায় তারটির দৈর্ঘ্য 4m। তারের উপাদানের ইয়াং এর গুণাঙ্ক 1.3x10^10 Nm-1। ভর সরিয়ে নিলে তারটির দৈর্ঘ্য কি পরিমাণ সংকুচিত হবে?

En_Q: A 15 kg mass hangs from a wire with a 2 mm^2 cross-section. With the mass hanging, the wire's length is 4m. The Young's modulus of the wire material is 1.3x10^10 Nm^{-1}. If the mass is removed, how much will the wire contract?

A. 0.0022m B. 0.0225m
C. 0.225m D. 2.25m

Topic: Mechanics
Question Type: Mathematical

Bangla B B B
English B B B

(a)

Bn_Q: অপবর্তন এক বিশেষ ধরনের—

En_Q: Diffraction is a special type of—

A. সমবর্তন (Polarization)
B. প্রতিফলন (Reflection)
C. ব্যাতিচার (Interference)
D. প্রতিসরণ (Refraction)

Topic: Wave Optics
Question Type: Theoretical

Bangla C C D
English C C C

(b)

Bn_Q: একই তাপমাত্রায় সিলেট অপেক্ষা কুয়াকাটায় বেশি অস্বস্তিকর বোধ হয় কারণ__
En_Q: At the same temperature; it feels more uncomfortable in Kuakata than in Sylhet because__
A. সমুদ্রপৃষ্ঠ হতে সিলেটের উচ্চতা কুয়াকাটার চেয়ে বেশি (Sylhet's altitude from sea level is higher than Kuakata's)
B. সমুদ্রপৃষ্ঠ হতে সিলেট বহুদূরে (Sylhet is far from the sea)
C. বিষুব রেখা হতে সিলেট বেশী দূরে (Sylhet is farther from the equator)
D. কোনোটিই নয় (None of these)

Topic: Thermodynamics
Question Type: Reasoning

Bangla A A A
English B B A

(c)

Bn_Q: 'M' ভর এবং 'a' প্রান্ত বিশিষ্ট একটি সমবাহু বর্গক্ষেত্রের একটি কর্ণের সাপেক্ষে এর জড়তার ভ্রামক কত?

En_Q: What is the moment of inertia of a square of mass 'M' and side 'a' about one of its diagonals?

A. $Ma^2/3$ B. $Ma^2/6$
C. $Ma^2/9$ D. $Ma^2/12$

Topic: Mechanics
Question Type: Theoretical

Bangla A D D
English D A A

(d)

Bn_Q: একটি পর্যবেক্ষণে দুইটি আলোর তরঙ্গের তীব্রতা যথাক্রমে 100 এবং 20 একক হলে, তাদের সমবায় তীব্রতার মান কত হবে?

En_Q: If two light waves have intensities of 100 and 20 units respectively, what will be the resultant intensity?

A. 120 B. 134
C. 345 D. 60

Topic: Wave Optics
Question Type: Mathematical

Bangla A B A
English A D A

(e)

Bn_Q: বাস্তবে একটি ইঞ্জিনের দক্ষতা সর্বোচ্চ কত পর্যন্ত পাওয়া যেতে পারে?

En_Q: In reality; what is the maximum possible efficiency of an engine?

A. 20% B. 25%
C. 50% D. 100%

Topic: Thermodynamics
Question Type: Theoretical

Bangla C C B
English B D D

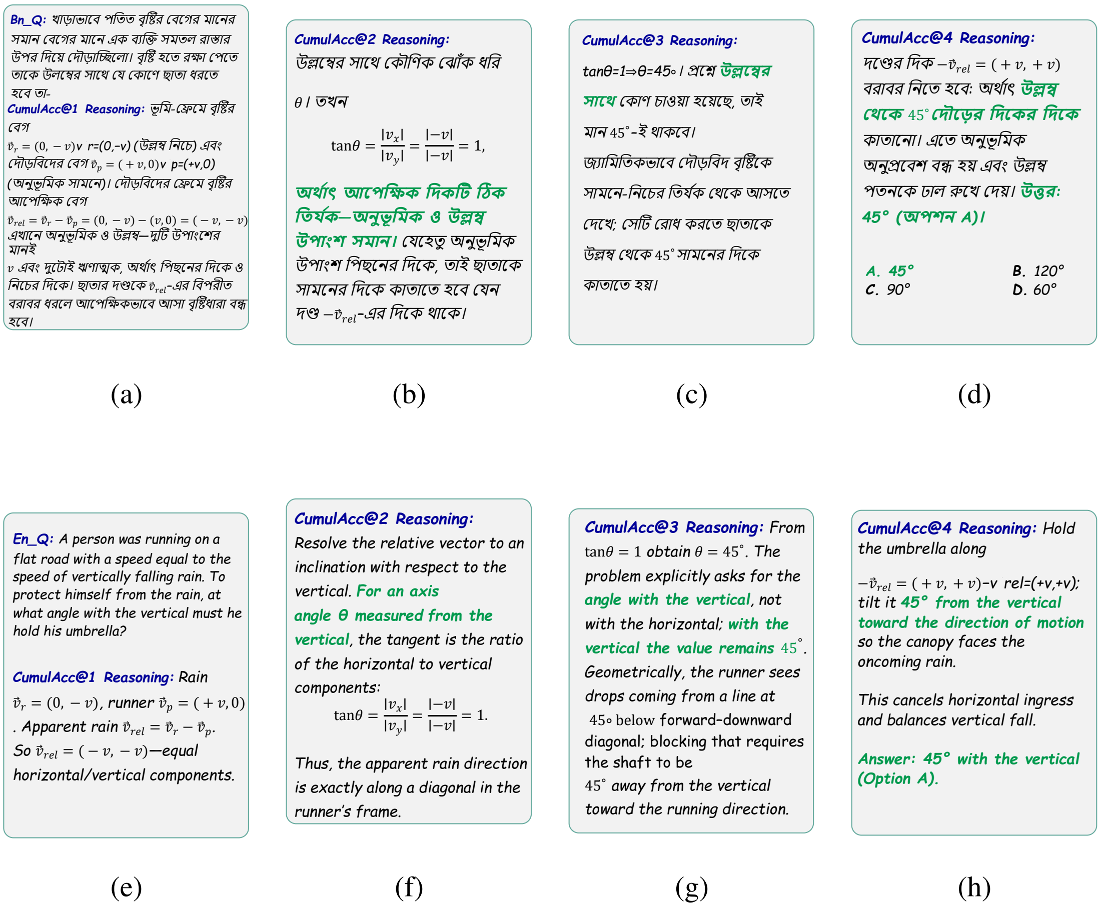
(f)

Figure 8: Representative error cases showing model performance on different question types. Each example displays the question in both Bangla and English, along with predictions from GPT-4.1, Gemini-2.5 Pro, and DeepSeek-R1 (left to right), with correct answers marked in green and incorrect ones marked in red.

Bangla fail in English and vice versa, strongly suggesting language-compartmentalized knowledge rather than unified multilingual understanding. Example (f) demonstrates Bangla superiority (2 of 3 correct) versus English failure (0 of 3 correct), indicating certain physics concepts are better preserved in low-resource language training despite smaller data volumes. These inconsistencies have critical implications for educational deployment: students querying the same problem in different languages may receive contradictory answers, undermining trust and pedagogical effectiveness. The patterns suggest current multilingual LLMs lack cross-lingual alignment mechanisms for domain-specific knowledge, necessitating explicit consistency constraints during training for reliable educational applications.

A.6 Cumulative Reasoning Improvements through Iterative Refinement

Beyond identifying failure patterns, our analysis reveals how DeepSeek-R1's cumulative accuracy mechanism systematically improves reasoning quality through iterative self-correction. Figures 9 and 10 demonstrate this progressive refinement across four iterations, where each cumulative pass adds verification layers that catch and correct errors from previous attempts. In the relative



Bn_Q: খাড়াভাবে পতিত বৃষ্টির বেগের মানের সমান বেগের মানে এক ব্যক্তি সমতল রাস্তার উপর দিয়ে দৌড়াচ্ছিলো। বৃষ্টি হতে রক্ষা পেতে তাকে উলম্বের সাথে যে কোণে ছাতা ধরতে হবে তা-

CumulAcc@1 Reasoning: ভূমি-ফ্রেমে বৃষ্টির বেগ

$\vec{v}_r = (0, -v)$v r=(0,-v) (উল্লম্ব নিচে) এবং দৌড়বিদের বেগ $\vec{v}_p = (+v, 0)$v p=(+v,0) (অনুভূমিক সামনে)। দৌড়বিদের ফ্রেমে বৃষ্টির আপেক্ষিক বেগ

$\vec{v}_{rel} = \vec{v}_r - \vec{v}_p = (0, -v) - (v, 0) = (-v, -v)$
এখানে অনুভূমিক ও উল্লম্ব—দুটি উপাংশের মানই

v এবং দুটোই ঋণাত্মক, অর্থাৎ পিছনের দিকে ও নিচের দিকে। ছাতার দণ্ডকে $\vec{v}_{rel}$-এর বিপরীত বরাবর ধরলে আপেক্ষিকভাবে আসা বৃষ্টিধারা বন্ধ হবে।

(a)

CumulAcc@2 Reasoning:
উল্লম্বের সাথে কৌণিক ঝোঁক ধরি

θ। তখন

$$\tan\theta = \frac{|v_x|}{|v_y|} = \frac{|-v|}{|-v|} = 1,$$

অর্থাৎ আপেক্ষিক দিকটি ঠিক তির্যক—অনুভূমিক ও উল্লম্ব উপাংশ সমান। যেহেতু অনুভূমিক উপাংশ পিছনের দিকে, তাই ছাতাকে সামনের দিকে কাতাতে হবে যেন দণ্ড $-\vec{v}_{rel}$-এর দিকে থাকে।

(b)

CumulAcc@3 Reasoning:

tanθ=1⇒θ=45∘। প্রশ্নে **উল্লম্বের সাথে** কোণ চাওয়া হয়েছে, তাই মান 45°-ই থাকবে।

জ্যামিতিকভাবে দৌড়বিদ বৃষ্টিকে সামনে-নিচের তির্যক থেকে আসতে দেখে; সেটি রোধ করতে ছাতাকে উল্লম্ব থেকে 45° সামনের দিকে কাতাতে হয়।

(c)

CumulAcc@4 Reasoning:
দণ্ডের দিক $-\vec{v}_{rel} = (+v, +v)$ বরাবর নিতে হবে: অর্থাৎ **উল্লম্ব থেকে 45° দৌড়ের দিকের দিকে** কাতানো। এতে অনুভূমিক অনুপ্রবেশ বন্ধ হয় এবং উল্লম্ব পতনকে ঢাল রুখে দেয়। **উত্তর: 45° (অপশন A)।**

A. 45° B. 120°
C. 90° D. 60°

(d)

En_Q: *A person was running on a flat road with a speed equal to the speed of vertically falling rain. To protect himself from the rain, at what angle with the vertical must he hold his umbrella?*

CumulAcc@1 Reasoning: *Rain $\vec{v}_r = (0, -v)$, runner $\vec{v}_p = (+v, 0)$. Apparent rain $\vec{v}_{rel} = \vec{v}_r - \vec{v}_p$. So $\vec{v}_{rel} = (-v, -v)$—equal horizontal/vertical components.*

(e)

CumulAcc@2 Reasoning:
Resolve the relative vector to an inclination with respect to the vertical. ***For an axis angle θ measured from the vertical****, the tangent is the ratio of the horizontal to vertical components:*

$$\tan\theta = \frac{|v_x|}{|v_y|} = \frac{|-v|}{|-v|} = 1.$$

Thus, the apparent rain direction is exactly along a diagonal in the runner's frame.

(f)

CumulAcc@3 Reasoning: *From* $\tan\theta = 1$ *obtain* $\theta = 45^\circ$. *The problem explicitly asks for the* ***angle with the vertical****, not with the horizontal;* ***with the vertical the value remains*** 45°. *Geometrically, the runner sees drops coming from a line at* 45∘ below *forward-downward diagonal; blocking that requires the shaft to be* 45° *away from the vertical toward the running direction.*

(g)

CumulAcc@4 Reasoning: *Hold the umbrella along*

$-\vec{v}_{rel} = (+v, +v)$*-v rel=(+v,+v); tilt it* ***45° from the vertical toward the direction of motion*** *so the canopy faces the oncoming rain.*

This cancels horizontal ingress and balances vertical fall.

Answer: 45° with the vertical (Option A).

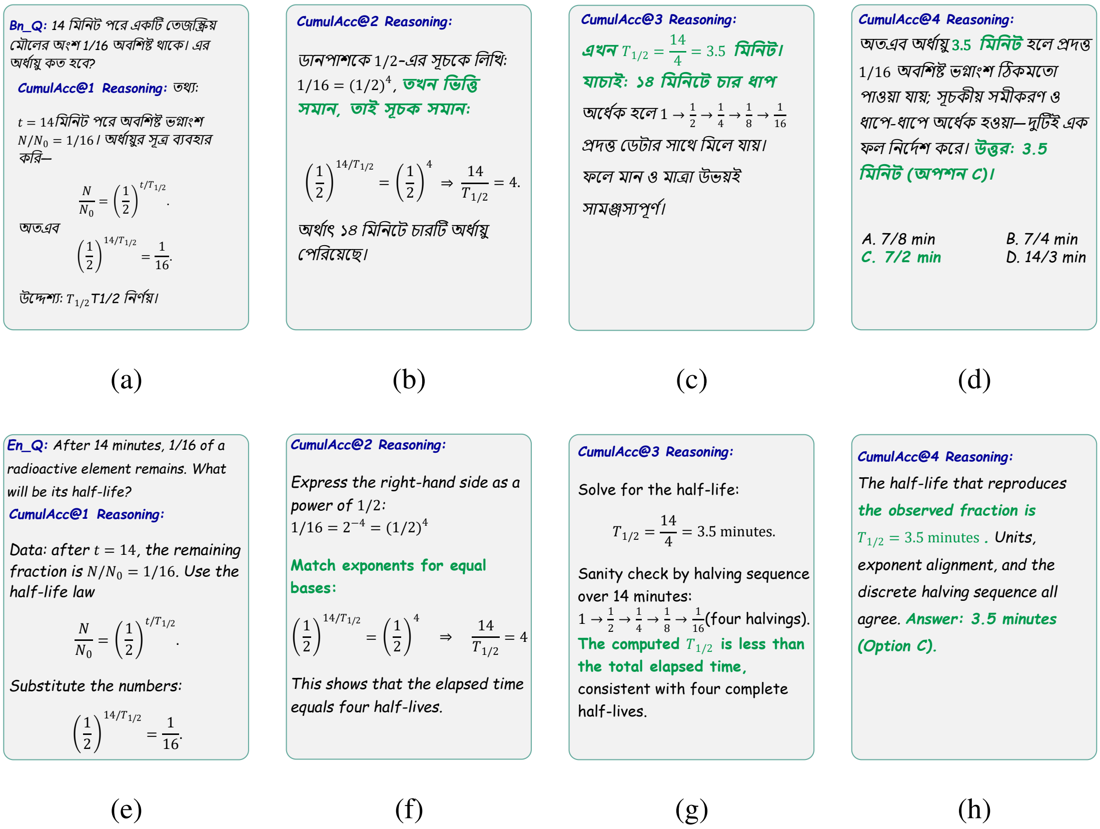
(h)

Figure 9: **DeepSeek-R1 iterative reasoning refinement for a relative motion problem.** Figure shows progressive reasoning across four cumulative accuracy steps for a problem on the umbrella angle needed to avoid rain at matching velocity (top: Bangla a–d, bottom: English e–h). Iterations (a,e) set velocity vectors, (b,f) compute relative velocity and tangent-based angles, (c,g) determine the 45° solution, and (d,h) verify via geometry (Option A). The parallel view highlights cross-lingual consistency in DeepSeek-R1's reasoning, with Bangla using detailed explanations and English favoring concise vector notation.

motion problem (Figure 9), CumulAcc@1 establishes the basic vector framework but lacks angular interpretation; CumulAcc@2 introduces tangent ratio calculations but doesn't fully resolve the geometric meaning; CumulAcc@3 explicitly derives $\theta = 45$ from $\tan\theta = 1$; and CumulAcc@4 provides complete geometric validation by confirming the umbrella orientation blocks the diagonal rain trajectory. This layered verification transforms incomplete initial reasoning into a robust, multi-perspective solution.

Similarly, the radioactive decay problem (Figure 10) showcases cumulative betterment: CumulAcc@1 sets up the exponential formula, CumulAcc@2 matches it to discrete half-life counts, CumulAcc@3 performs the arithmetic calculation with sanity checks, and CumulAcc@4 validates dimensional consistency across exponential and sequential interpretations. The progression from 48.00% (CumulAcc@1) to 77.71% (CumulAcc@4) in Bangla—a remarkable +29.71% absolute improvement—demonstrates that iterative refinement doesn't merely retry the same approach but systematically builds conceptual scaffolding, catching algebraic errors, verifying physical intuition, and ensuring cross-validation between mathematical formalism and discrete physical processes. This cumulative mechanism proves particularly effective for complex multi-step physics problems where single-pass reasoning often misses subtle conceptual connections or calculation errors.

The cross-lingual consistency observed across iterations (Figures 9e-h and 10e-h) further validates that these improvements stem from genuine reasoning refinement rather than language-specific pattern matching, as DeepSeek-R1 achieves similar



Bn_Q: 14 মিনিট পরে একটি তেজস্ক্রিয় মৌলের অংশ 1/16 অবশিষ্ট থাকে। এর অর্ধায়ু কত হবে?

CumulAcc@1 Reasoning: তথ্য:

$t = 14$মিনিট পরে অবশিষ্ট ভগ্নাংশ $N/N_0 = 1/16$। অর্ধায়ুর সূত্র ব্যবহার করি—

$$\frac{N}{N_0} = \left(\frac{1}{2}\right)^{t/T_{1/2}}.$$

অতএব

$$\left(\frac{1}{2}\right)^{14/T_{1/2}} = \frac{1}{16}.$$

উদ্দেশ্য: $T_{1/2}$T1/2 নির্ণয়।

(a)

CumulAcc@2 Reasoning:

ডানপাশকে 1/2-এর সূচকে লিখি: $1/16 = (1/2)^4$, **তখন ভিত্তি সমান, তাই সূচক সমান:**

$$\left(\frac{1}{2}\right)^{14/T_{1/2}} = \left(\frac{1}{2}\right)^4 \Rightarrow \frac{14}{T_{1/2}} = 4.$$

অর্থাৎ ১৪ মিনিটে চারটি অর্ধায়ু পেরিয়েছে।

(b)

CumulAcc@3 Reasoning:

এখন $T_{1/2} = \frac{14}{4} = 3.5$ **মিনিট।**

যাচাই: ১৪ মিনিটে চার ধাপ

অর্ধেক হলে $1 \rightarrow \frac{1}{2} \rightarrow \frac{1}{4} \rightarrow \frac{1}{8} \rightarrow \frac{1}{16}$

প্রদত্ত ডেটার সাথে মিলে যায়।

ফলে মান ও মাত্রা উভয়ই সামঞ্জস্যপূর্ণ।

(c)

CumulAcc@4 Reasoning:

অতএব অর্ধায়ু 3.5 **মিনিট** হলে প্রদত্ত 1/16 অবশিষ্ট ভগ্নাংশ ঠিকমতো পাওয়া যায়; সূচকীয় সমীকরণ ও ধাপে-ধাপে অর্ধেক হওয়া—দুটিই এক ফল নির্দেশ করে। **উত্তর: 3.5 মিনিট (অপশন C)।**

A. 7/8 min B. 7/4 min
C. 7/2 min D. 14/3 min

(d)

En_Q: After 14 minutes, 1/16 of a radioactive element remains. What will be its half-life?

CumulAcc@1 Reasoning:

Data: after $t = 14$, the remaining fraction is $N/N_0 = 1/16$. Use the half-life law

$$\frac{N}{N_0} = \left(\frac{1}{2}\right)^{t/T_{1/2}}.$$

Substitute the numbers:

$$\left(\frac{1}{2}\right)^{14/T_{1/2}} = \frac{1}{16}.$$

(e)

CumulAcc@2 Reasoning:

Express the right-hand side as a power of $1/2$:
$1/16 = 2^{-4} = (1/2)^4$

Match exponents for equal bases:

$$\left(\frac{1}{2}\right)^{14/T_{1/2}} = \left(\frac{1}{2}\right)^4 \quad \Rightarrow \quad \frac{14}{T_{1/2}} = 4$$

This shows that the elapsed time equals four half-lives.

(f)

CumulAcc@3 Reasoning:

Solve for the half-life:

$$T_{1/2} = \frac{14}{4} = 3.5 \text{ minutes.}$$

Sanity check by halving sequence over 14 minutes:
$1 \rightarrow \frac{1}{2} \rightarrow \frac{1}{4} \rightarrow \frac{1}{8} \rightarrow \frac{1}{16}$(four halvings).
The computed $T_{1/2}$ is less than the total elapsed time, consistent with four complete half-lives.

(g)

CumulAcc@4 Reasoning:

The half-life that reproduces **the observed fraction is** $T_{1/2} = 3.5$ minutes . Units, exponent alignment, and the discrete halving sequence all agree. **Answer: 3.5 minutes (Option C).**

(h)

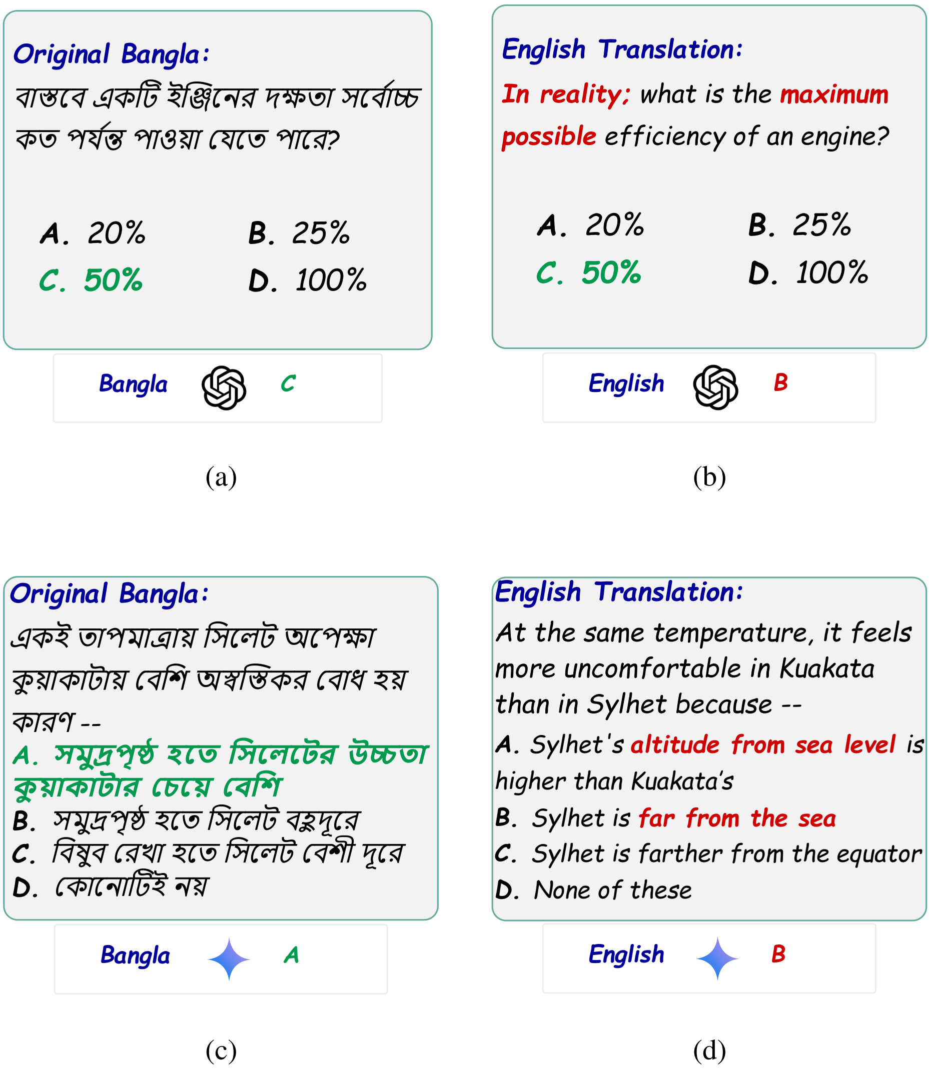
Figure 10: **DeepSeek-R1 iterative reasoning refinement for a radioactive decay problem.** Figure shows progressive reasoning across four cumulative accuracy steps for a half-life problem where 1/16 of a radioactive element remains after 14 minutes (top: Bangla a–d, bottom: English e–h). Iterations (a,e) set up the decay formula, (b,f) express 1/16 as $(1/2)^4$, (c,g) compute $T_{1/2} = 3.5$ min, and (d,h) verify dimensional consistency (Option C). This illustrates DeepSeek-R1's systematic decomposition of exponential decay problems into verifiable steps.

cumulative gains in English (62.67% to 79.50%, +16.83%). The success of this iterative approach suggests that educational AI systems should prioritize multi-pass reasoning architectures over single-shot prediction, enabling students to see how expert problem-solving progressively builds from foundations to verified solutions.

B Translation-Induced Evaluation Failures

Our analysis reveals systematic cross-lingual inconsistencies where translation artifacts alter model interpretations. Figure 11 illustrates two physics problems in which GPT-4.1 and Gemini-2.5 Pro provide different answers across Bangla and English versions due to semantic degradation during translation. In the first case, concerning maximum real engine efficiency (correct answer: 50%), GPT-4.1 correctly identifies the Bangla version's practical engineering framing and selects the right answer. However, in the English translation, the phrase "In reality" introduces semantic ambiguity, creating tension between practical and theoretical interpretations. As a result, GPT-4.1 shifts its reasoning toward typical operational ranges (25%) rather than engineering limits. In the second case, regarding why Kuakata feels more uncomfortable than Sylhet at equal temperatures (correct answer: A, based on altitude affecting atmospheric pressure and evaporative cooling), the Bangla version presents a truncated, obviously incorrect distractor, allowing Gemini-2.5 Pro to select the correct option. The English translation completes this distractor but alters its semantics, making it appear plausible. Consequently, Gemini-2.5 Pro chooses the wrong answer in English, despite understanding the physics correctly in Bangla.

These examples demonstrate that translation-induced failures can occur at multiple linguistic levels. Question-stem translations may introduce contextual ambiguity, as in the engine efficiency example, while option-level translation quality can



Original Bangla:
বাস্তবে একটি ইঞ্জিনের দক্ষতা সর্বোচ্চ কত পর্যন্ত পাওয়া যেতে পারে?

A. 20% B. 25%
C. 50% D. 100%

(a)

English Translation:
In reality; what is the maximum possible efficiency of an engine?

A. 20% B. 25%
C. 50% D. 100%

(b)

Original Bangla:
একই তাপমাত্রায় সিলেট অপেক্ষা কুয়াকাটায় বেশি অস্বস্তিকর বোধ হয় কারণ --
A. সমুদ্রপৃষ্ঠ হতে সিলেটের উচ্চতা কুয়াকাটার চেয়ে বেশি
B. সমুদ্রপৃষ্ঠ হতে সিলেট বহুদূরে
C. বিষুব রেখা হতে সিলেট বেশী দূরে
D. কোনোটিই নয়

(c)

English Translation:
At the same temperature, it feels more uncomfortable in Kuakata than in Sylhet because --
A. Sylhet's altitude from sea level is higher than Kuakata's
B. Sylhet is far from the sea
C. Sylhet is farther from the equator
D. None of these

(d)

Figure 11: Figure illustrates translation-induced evaluation failures in GPT-4.1 and Gemini-2.5 Pro. (a,b) Engine efficiency: GPT-4.1 answers correctly in Bangla (C, 50%) but selects B (25%) in English due to semantic ambiguity ("In reality" vs. "maximum possible"). (c,d) Thermodynamic comfort: Gemini-2.5 Pro selects A in Bangla but B in English because option translation alters meaning. Green marks correct answers; red highlights translation-induced faults.

change distractor plausibility, as seen in the thermodynamic comfort case. Importantly, aggregate performance metrics mask these failures: both models maintain high overall accuracy (GPT-4.1: 89.52% Bangla, 89.90% English; Gemini-2.5 Pro: 89.52% Bangla, 87.25% English), yet individual question analysis reveals systematic cross-lingual interpretation divergences. This has critical implications for multilingual educational AI, where students working in different languages may face substantively different reasoning challenges even on ostensibly identical problems.

Based on these findings, we recommend several practices for multilingual benchmark construction: preserving contextual precision across languages, validating semantic equivalence at the option level to avoid inadvertently changing distractor quality, analyzing cross-lingual response patterns to flag potential translation issues, and incorporating expert review protocols to ensure both question stems and options maintain the intended meaning. Figure 11 highlights that even without errors in answer key mapping, translation-altered semantics can undermine evaluation reliability, calling for careful consideration in the design of cross-lingual benchmarks.

C System Prompts

Single Inference (Cumul Acc@1). For initial evaluation, we used the following system prompt:

System Prompt for Bangla Physics MCQ (Single Inference)

You are a world-class Physics teacher specialized in solving MCQs in Bangla.
Your role:

- Read the question carefully. Questions may be in Bangla with English symbols, formulas, or expressions mixed in.
- Identify the topic (e.g., Mechanics, Thermodynamics, Wave Optics, Electromagnetism, Quantum Physics, Modern Physics).
- Identify the question type (Mathematical, Theoretical, Definition, Reasoning, Application, Miscellaneous).
- Work through the problem step by step, showing reasoning and calculations in Bangla where natural, but keep math notation in standard English symbols.
- Always check units, conversions, and scientific notation.
- Be concise but accurate in reasoning.
- At the end, give the correct option as a single letter: A, B, C, or D — nothing else on the final line.

Iterative Re-evaluation (Cumul Acc@2-4). For subsequent attempts on incorrectly answered questions, we modified the prompt to encourage deeper analysis:

System Prompt for Bangla Physics MCQ (Iterative Re-evaluation)

You are a world-class Physics teacher specialized in solving MCQs in Bangla.
Your role:

- Analyze each question deeply and answer carefully, rechecking the reasoning step by step, rethink and answer wisely.
- Read the question carefully. Questions may include English text mixed with formulas, symbols, or expressions.
- Identify the topic (e.g., Mechanics, Thermodynamics, Wave Optics, Electromagnetism, Quantum Physics, Modern Physics).
- Identify the question type (Mathematical, Theoretical, Definition, Reasoning, Application, Miscellaneous).
- Work through the problem step by step, showing reasoning and calculations.
- Always check units, conversions, and scientific notation.
- Be concise but accurate in reasoning.
- At the end, provide the correct option as a single letter: A, B, C, or D — nothing else on the final line.

C.1 Prompts for English Question Answering

Single Inference (Cumul Acc@1).

System Prompt for English Physics MCQ (Single Inference)

You are a world-class Physics teacher specialized in solving MCQs in English.
Your role:

- Read the question carefully.
- Identify the topic (e.g., Mechanics, Thermodynamics, Wave Optics, Electromagnetism, Quantum Physics, Modern Physics).
- Identify the question type (Mathematical, Theoretical, Definition, Reasoning, Application, Miscellaneous).
- Work through the problem step by step, showing reasoning and calculations clearly, keeping math notation in standard English symbols.
- Always check units, conversions, and scientific notation.
- Be concise but accurate in reasoning.
- At the end, give the correct option as a single letter: A, B, C, or D — nothing else on the final line.

Iterative Re-evaluation (Cumul Acc@2-4).

System Prompt for English Physics MCQ (Iterative Re-evaluation)

You are a world-class Physics teacher specialized in solving MCQs in English.
Your role:

- Analyze each question deeply and answer carefully, rechecking the reasoning step by step, rethink and answer wisely.
- Read the question carefully.
- Identify the topic (e.g., Mechanics, Thermodynamics, Wave Optics, Electromagnetism, Quantum Physics, Modern Physics).
- Identify the question type (Mathematical, Theoretical, Definition, Reasoning, Application, Miscellaneous).
- Work through the problem step by step, showing reasoning and calculations clearly.
- Always check units, conversions, and scientific notation.
- Be concise but accurate in reasoning.
- At the end, provide the correct option as a single letter: A, B, C, or D — nothing else on the final line.

C.2 Translation Prompt for Dataset Creation

For creating the English version of our dataset, we used Gemini-2.5 Pro with manual verification. The translation prompt was designed to preserve mathematical expressions and technical terminology:

Prompt for Bangla-to-English Translation

You are a professional translator.
Task: Translate ONLY Bangla text into English, keep English text unchanged.
Input format (comma-separated):
id,question,A,B,C,D,answer
Output rules:

- Output must use commas (,) as the column separator.
- Exactly 7 columns: id,question,A,B,C,D,answer
- In the question column: NEVER output commas. If the translation needs a comma, replace it with a semicolon (;) instead.
- Do not add extra columns.
- Do not add explanations, comments, or metadata.
- Keep numbers, math expressions, and symbols unchanged.
- Leave English text unchanged if it already exists.

C.3 Topic and Question Type Categorization Prompt

For systematic categorization of questions across physics domains and cognitive complexity levels, we used Gemini-2.5 Pro with the following prompt:

Prompt for Topic and Question Type Classification

You are a Physics exam classifier.
Task: Analyze each question (and optionally the options and correct answer) to determine:

1. The most relevant Physics topic from the allowed list.
2. The most appropriate Question Type from the allowed list.

Allowed Topics:
Electromagnetism, Mechanics, Thermodynamics, Wave Optics, Quantum Physics, Modern Physics
Allowed Question Types:
Mathematical, Theoretical, Definition, Reasoning, Application, Miscellaneous
Output Instructions:

- Input is CSV with columns: id,question,A,B,C,D,answer
- For each row, return ONLY: id,topic,question_type
- One line per question
- No extra text, commentary, or explanations
- If unsure, use Miscellaneous for both topic and question_type

Example Input:
1,Diffraction is a special type of—,Polarization,Reflection,Interference,Refraction,C
2,If the momentum of an object of mass 50 kg is 200 kgms-1; its kinetic energy will be-,200 J,300 J,400 J,500 J,C
Example Output:
1,Wave Optics,Theoretical
2,Mechanics,Mathematical

LP-FT-LoRA : A Three-Stage PEFT Framework for Efficient Domain Adaptation in Bangla NLP Tasks

Tasnimul Hossain Tomal[1], **Anam Borhan Uddin**[1], **Intesar Tahmid**[1], **Mir Sazzat Hossain**[1, 2], **Md Fahim**[1, 2, †], **Md Farhad Alam**[1]

[1]*Penta Global Limited* [2]*Center for Computational & Data Sciences*
[†] *Project Lead*

Correspondence: {tomal.tasnimul,fahimcse381}@gmail.com

Abstract

Adapting large pre-trained language models (LLMs) to downstream tasks typically requires fine-tuning, but fully updating all parameters is computationally prohibitive. Parameter-Efficient Fine-Tuning (PEFT) methods like Low-Rank Adaptation (LoRA) reduce this cost by updating a small subset of parameters. However, the standard approach of jointly training LoRA adapters and a new classifier head from a cold start can lead to training instability, as the classifier chases shifting feature representations. To address this, we propose LP-FT-LoRA , a novel three-stage training framework that decouples head alignment from representation learning to enhance stability and performance. Our framework first aligns the classifier head with the frozen LM backbone via linear probing, then trains only the LoRA adapters to learn task-specific features, and finally performs a brief joint refinement of the head and adapters. We conduct extensive experiments on five Bangla NLP benchmarks across four open-weight compact transformer models. The results demonstrate that LP-FT-LoRA consistently outperforms standard LoRA fine-tuning and other baselines, achieving state-of-the-art average performance and showing improved generalization on out-of-distribution datasets. Code for this paper is available at https://github.com/tomal66/lp-ft-lora.

1 Introduction

The paradigm of pre-training and fine-tuning has become the de-facto standard for natural language processing (NLP), with large language models (LLMs) based on the Transformer architecture (Devlin et al., 2019) demonstrating remarkable capabilities across a wide array of tasks (Zhao et al., 2024). To adapt these powerful but general-purpose models to specific downstream applications, fine-tuning is essential. However, updating all the parameters of a multi-billion parameter model is computationally extensive, requiring substantial memory and GPU resources. This has spurred the development of Parameter-Efficient Fine-Tuning (PEFT) methods.

PEFT techniques aim to adapt LLMs by updating only a small fraction of their total parameters, drastically reducing the computational burden while often matching or even exceeding the performance of full fine-tuning. Among these, Low-Rank Adaptation (LoRA) (Hu et al., 2022) has emerged as a particularly effective and widely adopted method. LoRA injects trainable low-rank matrices into the model's layers, allowing for efficient task-specific adaptation without modifying the original pre-trained weights.

Concurrently, linear probing remains a canonical and lightweight transfer learning protocol (Kumar et al., 2022). In this approach, the entire pre-trained LM backbone is frozen, and only a newly added classification head is trained. While fast and memory-efficient, its success hinges on the strong assumption that the downstream task's classes are already linearly separable in the model's frozen feature space. This assumption often breaks down under significant domain shifts, limiting its effectiveness for more complex adaptation scenarios.

Although both LoRA and linear probing are powerful, they present distinct challenges when applied in isolation. Standard LoRA fine-tuning, which typically involves jointly training the LoRA adapters (ϕ_{LoRA}) and a randomly initialized classifier head (ϕ_C), can suffer from training instability. The classifier head must learn to interpret features that are themselves being modified, a "moving target" problem that can lead to noisy gradients and slow convergence (Rajput and Mehta, 2025). On the other hand, linear probing's inability to adapt the backbone's representations makes it unsuitable for tasks where the pre-trained features are insufficient.

In this paper, we identify and address a critical gap: the need for a framework that systematically stabilizes the fine-tuning process while en-

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 212–222
December 23, 2025 ©2025 Association for Computational Linguistics

abling robust representation learning. We propose **LP-FT-LoRA** , a novel three-stage fine-tuning framework that explicitly decouples classifier head alignment from adapter-based representation learning. LP-FT-LoRA mitigates the instabilities of standard LoRA fine-tuning and overcomes the representational rigidity of simple linear probing. Our contributions are threefold:

- We introduce LP-FT-LoRA , a novel three-stage training framework that synergistically combines linear probing and LoRA for efficient and stable adaptation of LLMs.
- Through extensive experiments on five Bengali NLP datasets and four different model architectures, we demonstrate that LP-FT-LoRA consistently outperforms standard LoRA fine-tuning and other strong baselines.
- We provide a detailed analysis of the framework's robustness to out-of-distribution data and conduct comprehensive ablation studies to dissect the impact of key hyperparameters.

The rest of this paper is organized as follows: Section 2 reviews related work, Section 3 provides preliminaries on the core techniques, Section 4 details our proposed method, Section 5 describes the experimental setup, Section 6 presents our results and analysis, and Section 7 concludes the paper.

2 Related Work

In this section, we situate our approach within prior work on linear probing, parameter-efficient fine-tuning, and multi-stage fine-tuning, highlighting how existing methods motivate and contrast with our proposed framework.

2.1 Linear Probing

Linear probing and fine-tuning are canonical transfer learning protocols, where staged approaches like LP-FT improve performance and preserve representations under distribution shifts (Tomihari and Sato, 2024; Kumar et al., 2022). Advances in linear probing evaluation have introduced more robust metrics and demonstrated its utility in separating evaluation contexts from deployment prompts (Thilak et al., 2024; Nguyen et al., 2025).

2.2 Parameter Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) methods, such as adapter modules and the widely adopted LoRA, adapt large models by updating a small fraction of parameters (Houlsby et al., 2019; Hu et al., 2022). Recent LoRA variants enhance performance by using different learning rates, dynamically allocating parameter budgets based on layer importance, or employing layer-wise adaptive rank allocation (Hayou et al., 2024; Mao et al., 2024; Gu et al., 2025). Other extensions explore multiplicative updates or combine quantization with adaptive rank selection for highly compressed models (Bihany et al., 2025; Kim et al., 2024). The theoretical underpinnings for these methods are provided by the delta-tuning framework, with recent insights also revealing benefits of non-zero initialization practices (Ding et al., 2023; Li et al., 2025).

2.3 Multi-stage Fine-tuning

Multi-stage fine-tuning has been explored through progressive frameworks that mitigate catastrophic forgetting and in continual learning settings that manage knowledge conflicts (Hou et al., 2024; Guan et al., 2025). Hybrid PEFT methods and advanced multi-task architectures enable uncertainty quantification and fine-grained task specialization (Chai et al., 2025; Ning et al., 2025). While linear probing and LoRA are each well studied, a single multi-stage framework that integrates linear probing with LoRA fine-tuning for domain-specialized classification appears to be unaddressed.

3 Preliminaries

This section formalizes the linear probing fine-tuning and LoRA fine-tuning protocols and delineates the research scope that motivates our proposed framework.

3.1 Linear Probing Fine-Tuning

Kumar et al. (Kumar et al., 2022) explored the theoretical foundations and operational mechanisms of linear probing, particularly in the context of out-of-distribution (OOD) tasks. Their study also compared the performance of linear probing and full fine-tuning. The experimental results demonstrated that while fine-tuning outperforms linear probing on in-distribution (ID) tasks, it struggles with generalization on OOD tasks. Based on these observations, the authors proposed a hybrid approach called *Linear Probing Fine-Tuning* (LP-FT).

Omitting theoretical details, the operational workflow of LP-FT is as follows. Given a pre-trained LM backbone denoted as ϕ_M, a classifier head ϕ_C is appended on top. In standard fine-tuning, the entire network $[\phi_{LM}, \phi_C]$ is jointly trained based on the loss from the downstream task. In contrast, linear probing keeps ϕ_{LM} frozen and trains only the classifier ϕ_C. LP-FT introduces a two-stage training strategy:

Stage 1 (Linear Probing): The backbone ϕ_{LM} is frozen, and only the classifier ϕ_C is trained using the downstream loss.

Stage 2 (Fine-Tuning): Both the backbone ϕ_{LM} and the previously trained classifier ϕ_C are jointly fine-tuned on the downstream task.

This staged approach utilizes the robustness of linear probing in the initial phase and the representational flexibility of fine-tuning in the later phase, resulting in improved generalization across both ID and OOD settings.

3.2 LoRA Fine-Tuning

Low-Rank Adaptation (LoRA) (Hu et al., 2022) is a Parameter-Efficient Fine-Tuning (PEFT) technique designed to reduce the computational and memory overhead associated with traditional fine-tuning of large pre-trained models. Rather than updating all model parameters, LoRA introduces a low-rank decomposition to capture task-specific adaptations while training a classifier head jointly for the target task. A visual architecture of the procedure is shown in Figure 1a.

Let the pre-trained model backbone be denoted as ϕ_{LM}, with its associated weight matrix $\mathbf{W}$. During LoRA fine-tuning, $\mathbf{W}$ remains frozen. Instead of directly updating $\mathbf{W}$, LoRA introduces a learnable, low-rank update matrix $\Delta\mathbf{W}$, defined as:

$$\Delta\mathbf{W} = \mathbf{A}\mathbf{B}^T$$

where $\mathbf{A} \in \mathbb{R}^{d\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times d}$, with $r \ll d$. This low-rank factorization ensures that the number of additional trainable parameters is significantly smaller than in full-rank updates. The matrices $\mathbf{A}$ and $\mathbf{B}$ encode the task-specific information required for adaptation, with $\mathbf{A}$ representing learned transformations across output dimensions and $\mathbf{B}$ across input dimensions. We denote the LoRA adapter parameters as $\phi_{LoRA} = \{\mathbf{A}, \mathbf{B}\}$.

The final adapted weight matrix $\mathbf{W}'$ used during inference is given by:

$$\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W} = \mathbf{W} + \mathbf{A}\mathbf{B}^T$$

In addition to the LoRA adapters ϕ_{LoRA}, a task-specific classifier head $\phi_C \in \mathbb{R}^{h\times C}$ is introduced, where h is the hidden dimension of the backbone and C is the number of classes. This classifier head is randomly initialized and trained jointly with ϕ_{LoRA}, enabling the model to learn both feature adaptations and classification mappings simultaneously during fine-tuning.

3.3 Research Scope

With the backbone ϕ_{LM} frozen in Linear Probing Fine-Tuning, training only the head ϕ_C implicitly assumes downstream classes are linearly separable in the pretrained feature space. Therefore, for under domain shift, this approach often fails, and ϕ_C can merely reweight insufficient features.

In the case of LoRA Fine-Tuning, jointly optimizing $\{\phi_{\text{LoRA}}, \phi_C\}$ from a cold start requires representation learning and classification. It induces noisy gradients and acute sensitivity to LoRA rank r and learning rate (Hayou et al., 2024). Furthermore, if ϕ_C is trained jointly from scratch, the head can chase moving features, which shifts ϕ_{LM}'s pretrained representations, resulting in slow convergence (Tomihari and Sato, 2024).

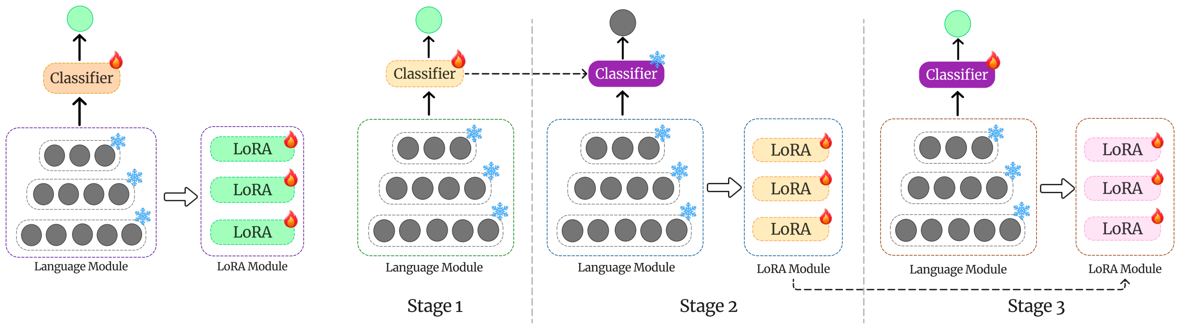
In our proposed framework LP-FT-LoRA , we mitigate the above-mentioned issues through a three-stage fine-tuning process. To the best of our knowledge, LP-FT-LoRA is the first framework to combine *linear probing* of the classifier (ϕ_C), *LoRA-only* probing of adapters (ϕ_{LoRA}) on a frozen backbone (ϕ_{LM}), and a brief joint refinement of $\phi_{\text{LoRA}}, \phi_C$ for explicit decoupling of head alignment from adapter representation learning.

4 Proposed Method: LP-FT-LoRA

In this work, we propose LP-FT-LoRA , a three-stage training framework that integrates LP-FT into a LoRA-augmented network. The objective is to enable efficient and effective adaptation to downstream tasks through structured, progressive training.

Let ϕ_{LM} represent the frozen pre-trained language model backbone. The LoRA-specific trainable parameters associated with this model are denoted as ϕ_{LoRA}, and the classifier head is represented by ϕ_C. The overall model architecture can thus be described as $[\phi_{LM}, \phi_{\text{LoRA}}, \phi_C]$. The overall design is visualized in Figure 1b. The training process proceeds in the following three stages:

Stage 1: Linear Probing. In the initial stage,



(a) Std. LoRA FT

(b) Proposed LP-FT-LoRA

Figure 1: Standard LoRA Fine-tuning and proposed LP-FT-LoRA architecture

the pretrained language model's backbone ϕ_{LM} remains frozen, and a classifier head ϕ_C is added on top. Only the classifier ϕ_C is trained using a downstream loss function—specifically, the cross-entropy loss in our classification setup. This step allows the classifier to adapt to the output space of the frozen backbone.

Stage 2: LoRA Linear Probing. In this stage, we initialize the LoRA parameters ϕ_{LoRA} with randomly initialized low-rank matrices $\mathbf{A}$ and $\mathbf{B}$, inserted into the backbone ϕ_{LM}. The previously trained classifier ϕ_C is retained, but both ϕ_{LM} and ϕ_C are kept frozen during this phase. Only the LoRA parameters are updated. The goal of this stage is to enable the LoRA layers to learn task-specific or domain-specific representations without modifying the backbone or classifier.

Stage 3: Fine-Tuning. In the final stage, we jointly train the LoRA parameters ϕ_{LoRA} and the classifier head ϕ_C, while continuing to keep the backbone ϕ_{LM} frozen. Crucially, the training resumes from the previously learned weights: ϕ_{LoRA} from Stage 2 and ϕ_C from Stage 1. This step refines both the adaptation layers and the classifier for improved performance on the downstream task.

5 Experimental Setup

This section details the datasets, model architectures, training configuration, and baseline methods used to evaluate LP-FT-LoRA .

5.1 Datasets

We evaluate LP-FT-LoRA across five Bangla NLP benchmarks covering fake news detection, sarcasm detection, sentiment analysis, and emotion recognition. The datasets are:

- **BanFakeNews** (Hossain et al., 2020): A fake news detection dataset containing approximately 48K authentic and 1K fake Bangla news articles across multiple categories. The task involves binary classification to determine whether a news article is authentic or fake.

- **Sarcasm Detection**: A Kaggle competition dataset[1] comprising around 50K news headlines labeled as either Sarcastic (1) or Not-Sarcastic (0).

- **SentNoB** (Islam et al., 2021): A sentiment analysis dataset of public comments collected from social media on news and videos, labeled as Positive, Negative, or Neutral. The dataset contains 13.5K training samples with 1.5K validation and 1.5K test samples across 13 different domains.

- **Emotion Detection** (Irtiza Tripto and Eunus Ali, 2018): A YouTube comments dataset for emotion classification with five categories: anger/disgust, joy, sadness, fear/surprise, and none. The dataset captures diverse emotional expressions in Bangla user-generated content.

- **Sentiment Classification** (Irtiza Tripto and Eunus Ali, 2018): A fine-grained sentiment analysis task using the same comment corpus as emotion detection, with five sentiment classes: Strongly Positive, Positive, Neutral, Negative, and Strongly Negative.

- **EmoNoBa** (Islam et al., 2022): A dataset for fine-grained, multi-label emotion analysis on

[1] https://www.kaggle.com/competitions/nlp-competition-cuet-ete-day-2022/data

noisy Bangla texts collected from social media. It includes labels for six basic emotions: joy, sadness, anger, disgust, fear, and surprise.

- **BanglaSarc** (Apon et al., 2022): A dataset for sarcasm detection in Bangla, compiled from comments on public Facebook posts. The task is a binary classification to identify text as either sarcastic or not sarcastic.

5.2 Model Architecture

We evaluate LP-FT-LoRA across four transformer-based open-weight models ranging from 360M to 1.5B parameters, covering compact to small scales. The selected backbones are: `SmolLM2-360M`, `Qwen3-0.6B`, `Gemma3-1B`, and `Qwen2.5-1.5B`. Their specifications are summarized in Table 1.

Model	**Params**	**Layers**	**Hidden**
SmolLM2-360M	360M	24	960
Qwen3-0.6B	600M	24	1024
Gemma3-1B	1B	18	2048
Qwen2.5-1.5B	1.5B	28	1536

Table 1: Architecture specifications of backbone models.

For adaptation, we apply LoRA with task-specific classifier heads that match hidden dimensions and output 2, 3, or 5 classes depending on the task.

5.3 Training Configuration

On each stage, we train with *AdamW* optimizer under a cosine learning rate schedule with a warm-up ratio of 0.03. We use a maximum sequence length of 512, a base learning rate of 2×10^{-4}, 4 training epochs, and a per-device batch size of 8 with gradient accumulation of 8 (effective batch size $8 \times 8 = 64$ sequences per update on a single device). We employ the SDPA attention path for training in this framework.

For LoRA, we use rank $r = 16$, targeting attention and MLP projections. The scaling factor α is 16 across all models, with a dropout rate of 0.05. All experiments are conducted on a single NVIDIA Tesla P100 (16 GB) GPU in the Kaggle environment. Implementations use Python 3.11 with PyTorch, Hugging Face `transformers`, `datasets`, `accelerate`, and `peft`.

5.4 Baseline Methods

We compare LP-FT-LoRA against the following baseline approaches:

- **Linear Probing (LP):** Training only the classifier head ϕ_C while freezing backbone ϕ_{LM}.
- **Standard LoRA Fine-Tuning:** Joint training of LoRA parameters ϕ_{LoRA} and classifier ϕ_C from random initialization.
- **LoRA Linear Probing:** Training of LoRA parameters ϕ_{LoRA} and keeping the classifier ϕ_C frozen.

6 Result and Analysis

This section presents the empirical results of LP-FT-LoRA and analyzes their implications across tasks, models, and baselines along with detailed ablation studies.

6.1 Performance of LP-FT-LoRA

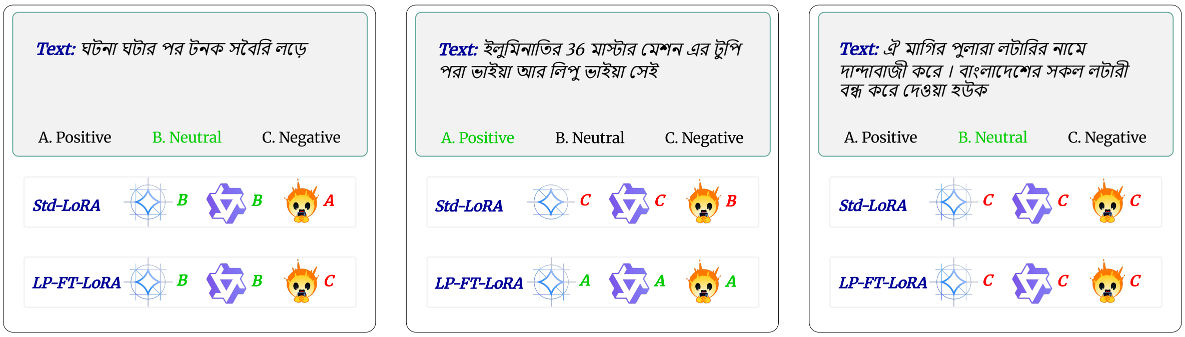
Analysis Across Datasets. As shown in Table 2, LP-FT-LoRA demonstrates consistent improvements across diverse task types. On the *BanFake* fake news detection dataset, LP-FT-LoRA achieves the best performance across all four backbone models, with accuracies ranging from 94.83% (SmolLM2-360M) to 98.82% (Gemma3-1B), outperforming Standard-LoRA-FT by 0.82–3.05 percentage points. For the Emotion classification task, LP-FT-LoRA consistently secures the top position on Qwen3-0.6B and Gemma3-1B, showing notable gains over Standard-LoRA-FT, with the most significant improvement observed on Qwen3-0.6B (58.48% vs. 52.25%). On SmolLM2-360M, LP-FT-LoRA achieves the second-best emotion score (50.19%), marginally behind Standard-LoRA-FT (51.03%). The Sarcasm detection task also favors LP-FT-LoRA , achieving best or tied-best results across all models with accuracies between 93.65% and 95.36%. In contrast, on the Sentiment classification task, LP-FT-LoRA exhibits more mixed results, achieving the best score on SmolLM2-360M (58.81%), while being slightly outperformed by Standard-LoRA-FT or LoRA Linear Probing on other models. This variability suggests that the three-stage training approach is particularly effective for tasks requiring nuanced semantic understanding and binary classification (emotion, fake news, sarcasm) but may offer diminishing returns on certain fine-grained multi-class sentiment distinctions.

Models	SentNoB	BanFake	Sarcasm	Emotion	Sentiment	Avg
Qwen3-0.6B						
Standard-LoRA-FT	69.99%	96.59%	94.37%	52.25%	**65.21%**	75.68%
Linear Probing	58.89%	87.66%	90.21%	42.91%	56.00%	67.13%
LoRA Linear Probing	68.10%	97.18%	94.41%	54.33%	64.87%	75.78%
LP-FT-LoRA	**71.31%**	**97.41%**	**94.54%**	**58.48%**	63.52%	**77.05%**
Gemma3-1B						
Standard-LoRA-FT	71.69%	95.77%	95.26%	52.94%	63.97%	75.93%
Linear Probing	63.18%	94.36%	91.96%	43.60%	55.44%	69.71%
LoRA Linear Probing	71.25%	98.47%	**95.36%**	53.98%	**65.10%**	76.83%
LP-FT-LoRA	**72.07%**	**98.82%**	**95.36%**	**55.02%**	63.75%	**77.00%**
SmolLM2 360M						
Standard-LoRA-FT	**67.91%**	93.15%	91.01%	**51.03%**	57.19%	72.06%
Linear Probing	50.44%	84.72%	88.37%	23.18%	45.23%	58.39%
LoRA Linear Probing	62.61%	92.48%	92.72%	38.64%	53.76%	68.04%
LP-FT-LoRA	65.70%	**94.83%**	**93.65%**	50.19%	**58.81%**	**72.64%**
Qwen2.5-1.5B						
Standard-LoRA-FT	**69.36%**	96.12%	94.45%	**56.10%**	62.74%	75.75%
Linear Probing	59.21%	86.49%	90.02%	32.18%	51.07%	63.79%
LoRA Linear Probing	68.60%	96.94%	94.37%	53.29%	**65.21%**	75.68%
LP-FT-LoRA	67.91%	**97.88%**	**94.68%**	56.06%	62.96%	**75.90%**

Table 2: Evaluation results (accuracy) on five datasets (SentNoB, BanFake, Sarcasm, Emotion, Sentiment) for four base models under different training strategies. **Bold** marks the best score within each model; underline marks the second-best.

Model-wise Comparison. Across all four backbone architectures, LP-FT-LoRA achieves the highest average accuracy: Qwen3-0.6B (77.05%), Gemma3-1B (77.00%), SmolLM2-360M (72.64%), and Qwen2.5-1.5B (75.90%), demonstrating its robustness across different model families and parameter scales. Notably, on Qwen3-0.6B and Gemma3-1B, LP-FT-LoRA outperforms Standard-LoRA-FT by 1.37 and 1.07 percentage points, respectively. On SmolLM2-360M, LP-FT-LoRA surpasses Standard-LoRA-FT by 0.58 percentage points (72.64% vs. 72.06%), while on Qwen2.5-1.5B, the improvement is 0.15 percentage points (75.90% vs. 75.75%). Compared to the two-stage LoRA Linear Probing approach, LP-FT-LoRA consistently delivers superior performance, with improvements of 0.17–4.60 percentage points, indicating that the additional fine-tuning stage (Stage 3) provides meaningful refinement. The gap between LP-FT-LoRA and Linear Probing is substantial across all models, underscoring the critical role of LoRA adaptation in the proposed framework.

Impact of Model Size. The experimental results reveal a non-linear relationship between model size and the effectiveness of LP-FT-LoRA . The 360M-parameter SmolLM2 achieves an average accuracy of 72.64%, while increasing model size to 600M (Qwen3-0.6B) yields a substantial 4.41-point improvement to 77.05%. Further scaling to 1B parameters (Gemma3-1B) provides only marginal changes (77.00%, essentially equivalent performance), and the 1.5B-parameter Qwen2.5 model achieves 75.90%, lower than the smaller Qwen3-0.6B and Gemma3-1B. This pattern suggests that LP-FT-LoRA is highly effective at extracting task-specific knowledge from compact models (600M-1B range), where the progressive training stages can efficiently leverage limited capacity. The diminishing returns at 1.5B parameters may indicate that larger models require different optimization strategies or that the chosen datasets reach a performance ceiling around 76-77% average accuracy, regardless of increased model capacity. Interestingly, Gemma3-1B and Qwen3-0.6B achieve nearly identical performance despite a significant parameter difference, suggesting that architectural design and pre-training quality may matter as much as raw model size for Bangla NLP tasks.


Text: ঘটনা ঘটার পর টনক সবৈরি লড়ে
A. Positive
B. Neutral
C. Negative
Std-LoRA
B
B
A
LP-FT-LoRA
B
B
C
Text: ইলুমিনাতির 36 মাস্টার মেশন এর টুপি পরা ভাইয়া আর লিপু ভাইয়া সেই
A. Positive
B. Neutral
C. Negative
Std-LoRA
C
C
B
LP-FT-LoRA
A
A
A
Text: ঐ মাগির পুলারা লটারির নামে দান্দাবাজী করে। বাংলাদেশের সকল লটারী বন্ধ করে দেওয়া হউক
A. Positive
B. Neutral
C. Negative
Std-LoRA
C
C
C
LP-FT-LoRA
C
C
C

(a) (b) (c)

Figure 2: Error analysis with Gemma3-1B, Qwen3-0.6B and SmolLM2-360M on three samples from the *SentNoB* dataset.

6.2 Error Analysis.

To assess LP-FT-LoRA 's effectiveness, we analyze three challenging examples from the *SentNoB* dataset using Gemma3-1B, Qwen3-0.6B, and SmolLM2-360M with both Standard LoRA (Std-LoRA) and LP-FT-LoRA . We identify two main error patterns: (i) confusion between similar sentiment classes and (ii) difficulty with sparse or conflicting sentiment cues.

The first example in Figure 2a shows both methods correctly predicting the label for straightforward text with clear polarity. This suggests LP-FT-LoRA maintains baseline performance on simple cases. In the second example (Figure 2b), Std-LoRA incorrectly predicts Negative/Neutral due to misleading named entities, while LP-FT-LoRA correctly identifies the Positive sentiment. This demonstrates LP-FT-LoRA 's improved handling of deceptive lexical cues.

The third example (Figure 2c) contains Bangla slang and strong language within a policy statement. The gold label is Neutral, but both methods predict Negative. This reveals a common bias where informal language and emphatic expressions override the actual discourse intent.

6.3 Ablation Studies

We ablate one hyperparameter at a time around a fixed configuration: LoRA rank r=16, LoRA scaling α=32, *Attn+MLP* target modules, batch size 8, learning rate 2×10^{-4}, and 4 epochs. The Macro-F1 scores are visualized in Figure 3a for LoRA and Figure 3b for training hyperparameters.

Impact of LoRA Hyperparameters

Rank. On *Sentiment*, performance peaks near the baseline r=16, with both lower (r=8) and higher (r=32) ranks underperforming. On *Emotion*, a smaller adaptation (r=8) is preferable, while larger ranks yield diminishing returns. Overall, moderate rank values are most reliable across datasets (Figure 3a).

Scaling factor α. Increasing α improves robustness. For *Sentiment*, α=64 delivers the strongest scores, substantially surpassing smaller values. *Emotion* also benefits from a higher scale, with α=64 slightly outperforming α=16 and α=32. This suggests that stronger LoRA scaling helps the adapter better fit both tasks when other settings are fixed (Figure 3a).

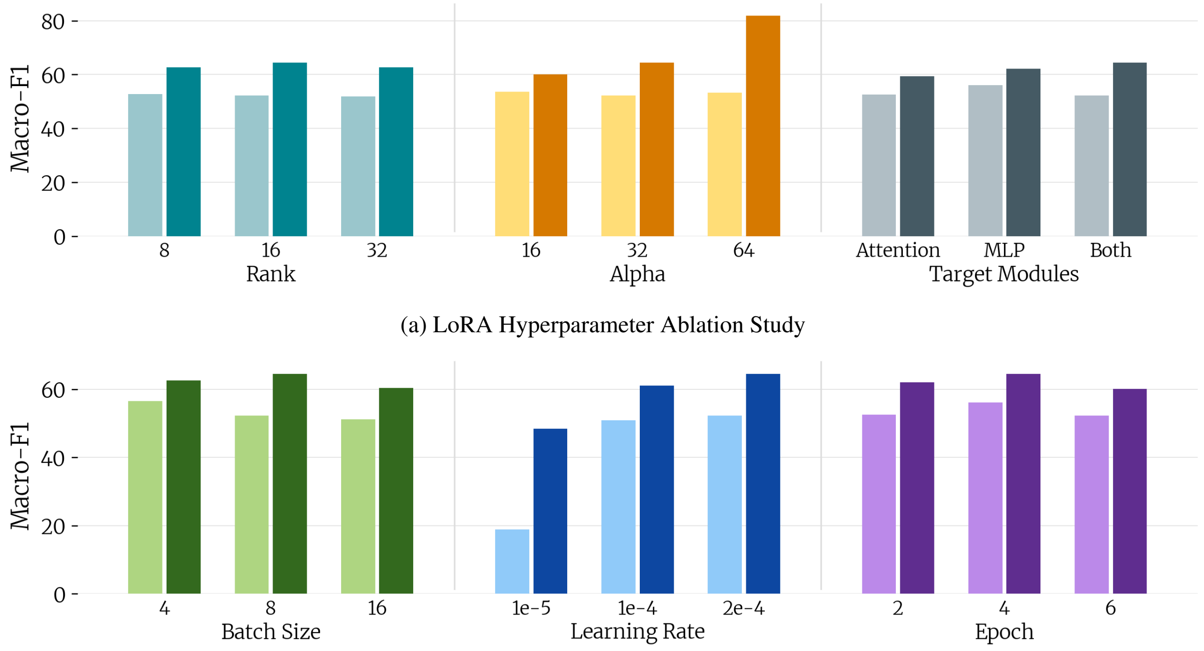
Target modules. Updating both *Attention* and *MLP* consistently outperforms targeting a single block on *Sentiment*. For *Emotion*, adapting only *MLP* edges out other choices, indicating dataset-specific sensitivities to where capacity is added. In practice, *Attention+MLP* is a safe default; *MLP*-only can be competitive for fine-grained tasks (Figure 3a).

Impact of Training Hyperparameters

Batch size. The baseline 8 works best for *Sentiment*, whereas a smaller batch (4) is preferable for *Emotion*. This mirrors the common trade-off that smaller batches can aid optimization on noisier, fine-grained tasks (Figure 3b).

Learning rate. A moderate step size is consistently favorable: 2×10^{-4} is optimal on both datasets, while too small harms performance (Figure 3b).

Training epochs. With other settings fixed, *Sentiment* peaks at 4 epochs and degrades with fewer or more updates, suggesting mild overfitting beyond



(a) LoRA Hyperparameter Ablation Study

(b) Training Hyperparameter Ablation Study

Figure 3: Hyperparameter ablation studies on Gemma3-1B. The darker bars represent the *Sentiment* dataset and lighter bars represent the *Emotion* dataset.

the optimum. *Emotion* benefits from a slightly longer schedule, with best results at 6 epochs. Tuning the stopping point remains important even for lightweight adapters (Figure 3b).

Across tasks, (i) moderate LoRA capacity ($r{\approx}8$-16) with higher scaling ($\alpha{\approx}64$) is effective; (ii) adapting both *Attention* and *MLP* is a strong default, though *MLP*-only can win on *Emotion*; and (iii) a mid-range training setup (batch 4-8, LR $2{\times}10^{-4}$, 4–6 epochs) consistently yields the best trade-off between stability and accuracy.

6.4 Cross-Dataset Evaluation

To evaluate the generalization ability of LP-FT-LoRA , we conduct cross-dataset experiments by training models on one dataset and directly testing, in a zero-shot manner, on another dataset from the same domain. We compare the performance of our proposed LP-FT-LoRA against Standard-LoRA fine-tuning across sentiment, emotion, and sarcasm domains using Gemma3-1B and Qwen3-0.6B backbones.

Sentiment. Transferring between the *Sentiment Classification* dataset and *SentNoB* demonstrates clear advantages for LP-FT-LoRA . When trained on Sentiment Classification and tested on SentNoB, LP-FT-LoRA achieves 55.55% (Gemma3-1B) and 51.70% (Qwen3-0.6B), surpassing Standard-LoRA by 1.33 and 2.20 percentage points, respectively. Conversely, when trained on *SentNoB* and tested on *Sentiment Classification*, LP-FT-LoRA yields stronger gains, improving by 4.37 points on Gemma3-1B and 1.68 points on Qwen3-0.6B.

Emotion. We examine transfer between *Emotion Detection* and *EmoNoBa*. Both models achieve nearly identical performance here, with Standard-LoRA slightly outperforming LP-FT-LoRA (34.98% vs. 34.67% for Gemma3-1B, and 35.87% vs. 35.85% for Qwen3-0.6B). This indicates that transferring across fine-grained emotion datasets remains a considerable challenge, and the incremental gain from progressive training is less pronounced compared to other domains.

Sarcasm. Cross-dataset transfer between the *Sarcasm Detection* dataset and *BanglaSarc* shows consistent improvements with LP-FT-LoRA . On Gemma3-1B, LP-FT-LoRA achieves 65.10% compared to 62.81% with Standard-LoRA, while on Qwen3-0.6B, it attains 52.27% versus 50.78%. These results highlight the effectiveness of LP-FT-LoRA in capturing transferable features for sarcasm recognition, which often relies on subtle pragmatic cues.

			Gemma3-1B		Qwen3-0.6B	
Domain	**Train Dataset**	**Test Dataset**	**LP-FT-LoRA**	**Std-LoRA**	**LP-FT-LoRA**	**Std-LoRA**
Sentiment	Sentiment	SentNoB	**55.55%**	54.22%	**51.70%**	49.50%
	SentNoB	Sentiment	**60.94%**	56.57%	**51.29%**	49.61%
Emotion	Emotion	EmoNoBa	34.67%	**34.98%**	35.85%	**35.87%**
Sarcasm	Sarcasm	Bangla Sarc	**65.10%**	62.81%	**52.27%**	50.78%

Table 3: Cross-dataset evaluation (accuracy). Models are trained on *Dataset 1* and evaluated on *Dataset 2*. Best score is in **bold**.

6.5 Computation and Training Time

Model	**Std LoRA**	**LP-FT LoRA**		
		S1	**S2**	**S3**
Qwen3-0.6B	2282	968	2181	2109
Gemma3-1B	2046	769	1861	1963
Qwen2.5-1.5B	4988	2403	4779	4767
SmolLM2-0.3B	2324	942	2144	2136

Table 4: Comparison of the average training time per epoch (in seconds) between Standard LoRA and LP-FT-LoRA across the previously mentioned dataset for various model architectures. Here S1 means Stage 1 and so on

The table 4 presents the average per-epoch training time (in seconds) for Standard LoRA and LP-FT-LoRA across three. The LP-FT LoRA method shows notable reductions in training time during Stage 1 (S1), where models such as Qwen3-0.6B and Gemma3-1B decrease from 2282 s to 968 s and from 2046 s to 769 s, respectively. Similar trends appear for SmolLM2-0.3B (2324 s → 942 s) and Qwen2.5-1.5B (4988 s → 2403 s). In later stages (S2 and S3), the training time approaches the Standard LoRA baseline, reflecting the increasing proportion of parameters being updated.

7 Conclusion

In this work, we introduced LP-FT-LoRA , a novel three-stage fine-tuning framework that integrates linear probing and LoRA to improve the adaptation of pre-trained language models for specialized classification tasks. Our approach methodically decouples classifier head alignment from adapter representation learning by first training the classifier on frozen features, then training the LoRA adapters while freezing the LM backbone and classifier, and finally jointly refining both components. This structured process is designed to mitigate the noisy gradients and slow convergence associated with standard end-to-end fine-tuning.

Our extensive experiments across four language models and five Bangla NLP datasets demonstrated that LP-FT-LoRA consistently outperforms standard LoRA fine-tuning and other strong baselines. While this study confirms the effectiveness of our method on Bangla classification tasks, future work could extend the framework to other languages, larger models, and different task formats, such as text generation. Further research could also explore dynamic stage transitions or integrate more advanced PEFT techniques to build upon these findings.

Limitations

This study has several limitations that should be acknowledged. The evaluation of the proposed LP-FT-LoRA framework is primarily conducted on medium-scale transformer models up to 1.5 billion parameters. Consequently, its effectiveness and scalability on larger models remain unverified, and different optimization strategies may be required for such settings.

The method has been tested exclusively on classification tasks within the Bangla language domain. While the results demonstrate strong performance gains, the generalizability of the approach to other languages or to different NLP tasks such as text generation has yet to be established. The multistage training process involves several hyperparameters and requires careful tuning specific to each dataset. This tuning complexity may limit out-of-the-box applicability and could introduce additional overhead in practical deployments.

Recent studies on transliteration and code-mixed datasets for Bangla are gaining increased attention (Fahim et al., 2024; Haider et al., 2024; Ahmed et al., 2024). It would be valuable to investigate how our model performs on these alternative text forms. Instead of focusing solely on standard Bangla datasets, future work could explore the model's effectiveness on transliterated and code-mixed Bangla data.

References

Fahim Ahmed, Md Fahim, Md Ashraful Amin, Amin Ahsan Ali, and AKM Rahman. 2024. Improving the performance of transformer-based models over classical baselines in multiple transliterated languages. In *ECAI 2024*, pages 4043–4050. IOS Press.

Tasnim Sakib Apon, Ramisa Anan, Elizabeth Antora Modhu, Arjun Suter, Ifrit Jamal Sneha, and MD Golam Rabiul Alam. 2022. Banglasarc: A dataset for sarcasm detection. *arXiv preprint arXiv:2209.13461*.

Harsh Bihany, Shubham Patel, and Ashutosh Modi. 2025. Lorma: Low-rank multiplicative adaptation for llms. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Yidong Chai, Yang Liu, Yonghang Zhou, Jiaheng Xie, and Daniel Dajun Zeng. 2025. A bayesian hybrid parameter-efficient fine-tuning method for large language models. *arXiv preprint arXiv:2508.02711*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pretrained language models. *Nature Machine Intelligence*, 5(3):220–235.

Md Fahim, Fariha Tanjim Shifat, Fabiha Haider, Deeparghya Dutta Barua, MD Sakib Ul Rahman Sourove, Md Farhan Ishmam, and Md Farhad Alam Bhuiyan. 2024. Banglatlit: A benchmark dataset for back-transliteration of romanized bangla. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14656–14672.

Jiancheng Gu, Jiabin Yuan, Jiyuan Cai, Xianfa Zhou, and Lili Fan. 2025. La-lora: Parameter-efficient fine-tuning with layer-wise adaptive low-rank adaptation. *Neural Networks*, 194:108095.

Changhao Guan, Chao Huang, Hongliang Li, You Li, Ning Cheng, Zihe Liu, Yufeng Chen, Jinan Xu, and Jian Liu. 2025. Multi-stage llm fine-tuning with a continual learning setting. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5484–5498.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, and Md Farhad Alam. 2024. Banth: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Md Zobaer Hossain, Md Ashraful Rahman, Md Saiful Islam, and Sudipta Kar. 2020. BanFakeNews: A dataset for detecting fake news in Bangla. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2862–2871, Marseille, France. European Language Resources Association.

Guiyang Hou, Yongliang Shen, and Weiming Lu. 2024. Progressive tuning: Towards generic sentiment abilities for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14545–14558.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Nafis Irtiza Tripto and Mohammed Eunus Ali. 2018. Detecting multilabel sentiment and emotions from bangla youtube comments. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–6.

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.

Khondoker Ittehadul Islam, Tanvir Hossain Yuvraz, Md Saiful Islam, and Enamul Hassan. 2022. EmoNoBa: A dataset for analyzing fine-grained emotions on noisy Bangla texts. 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 128–134, Online only. Association for Computational Linguistics.

Minsoo Kim, Sihwa Lee, Wonyong Sung, and Jungwook Choi. 2024. Ra-lora: Rank-adaptive parameter-efficient fine-tuning for accurate 2-bit quantized large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15773–15786.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*.

Shiwei Li, Xiandi Luo, Xing Tang, Haozhao Wang, Hao Chen, Weihong Luo, Yuhua Li, Xiuqiang He, and Ruixuan Li. 2025. Beyond zero initialization: Investigating the impact of non-zero initialization on lora fine-tuning dynamics. *arXiv preprint arXiv:2505.23194*.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11662–11675.

Jathan Nguyen et al. 2025. Probing and steering evaluation awareness of language models. *arXiv preprint arXiv:2507.01786*.

Lin Ning, Harsh Lara, Meiqi Guo, and Abhinav Rastogi. 2025. Mode: Effective multi-task parameter efficient fine-tuning with a mixture of dyadic experts. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

Sanjay Rajput and Priya Mehta. 2025. On the instability of jointly fine-tuning adapters and classification heads in large language models. *Journal of Machine Learning Research*, 26(45):1–25.

Vimal Thilak, Chen Huang, Omid Saremi, Laurent Dinh, Hanlin Goh, Preetum Nakkiran, Joshua M. Susskind, and Etai Littwin. 2024. Lidar: Sensing linear probing performance in joint embedding ssl architectures. In *International Conference on Learning Representations*.

Akiyoshi Tomihari and Issei Sato. 2024. Understanding linear probing then fine-tuning language models from NTK perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Wei Zhao, Li Chen, and Jin-seo Park. 2024. The new era of foundation models: A survey on pre-training, fine-tuning, and adaptation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 1–18.

Human–LLM Benchmarks for Bangla Dialect Translation: Sylheti & Chittagonian on the BanglaCHQ-Summ Corpus

Nowshin Mahjabin[1,*], Ahmed Shafin Ruhan[1,*], Mehreen Hossain Chowdhury[1,*], Md Fahim[2,3,†], Md. Azam Hossain[1,†]

[1] *Islamic University of Technology*
[2] *Center for Computational & Data Sciences* [3] *Penta Global Limited*
[*] Equal Contribution [†] Project Lead

Correspondence: {nowshin,ahmedshafin,mehreenhossain}@iut-dhaka.edu

Abstract

Millions in Bangladesh speak Sylheti and Chittagonian (Chatgaiyya) dialects, yet most public health guidance exists only in Standard Bangla, which creates barriers and safety risks. Ad-hoc translation further harms comprehension, while challenges such as scarce data, non-standard spelling, medical terms, numerals, and idioms make accurate translation difficult. We present **BanglaCHQ-Prantik**, the first benchmark for this setting, extending BanglaCHQ-Summ with human gold references from 17 native translators. We evaluate Qwen 2.5 3B, Gemma 3 1B, GPT-4o mini, and Gemini 2.5 Flash under zero-shot, one-shot, five-shot, and chain-of-thought prompts, using BLEU, ROUGE-1/2/L, and METEOR. Closed-source models (GPT-4o, Gemini 2.5) lead overall, with Gemini 2.5 Flash being strongest. Few-shot prompting helps especially for Sylheti, though errors persist with terminology, numerals, and idioms. The dataset is designed to support both NLP research and public health communication by enabling reliable translation across regional Bangla dialects. To our knowledge, this is the first medical-domain dataset for Sylheti/Chittagonian.

1 Introduction

Access to clear and reliable health information is critical for safe and effective care. However, in Bangladesh, most official and digital health materials are available only in Standard Bangla (Directorate General of Health Services (DGHS), 2021, 2022; Ministry of Health and Family Welfare (MOHFW), 2024). This poses significant barriers for millions who primarily speak regional dialects such as Sylheti and Chittagonian, spoken by approximately 11 million and 13 million people respectively (syl, 2025; chi, 2025). In the absence of linguistically accessible resources, patients often rely on informal translation or assistance from family members. These practices heighten the risk of miscommunication and medical errors (Al Shamsi et al., 2020). Enabling medical communication in regional dialects is therefore essential to ensure equitable access to health information across the population. Developing accurate dialectal medical translation is very challenging. Parallel resources for Sylheti and Chatgaiyya are scarce, spelling and syntax vary widely, idioms diverge from Standard Bangla and medical queries involve complex terminology, numerals, and dosage expressions. Such factors often lead to translation errors even in strong Machine Translation or Large Language Models (LLMs). While LLMs show promise with few-shot prompting, their ability in low-resource, medical-domain dialectal translation remains underexplored. Previous Bangla NLP studies have focused on related areas such as consumer-health question summarization (BanglaCHQ-Summ) (Khan et al., 2023) and general dialect-to-standard translation (ONUBAD, ChatgaiyyaAlap) (Sultana et al., 2025; Chowdhury et al., 2025) but none address the medical domain or provide benchmarks for dialectal evaluation. We bridge this gap by extending BanglaCHQ-Summ with human-verified Sylheti and Chatgaiyya translations, ensuring semantic fidelity and clinical accuracy. Using this dataset, we benchmark instruction-tuned LLMs Qwen 2.5 3B, Gemma 3 1B, GPT-4o mini, and Gemini 2.5 Flash under zero-shot, one-shot, five-shot, and chain-of-thought prompting, evaluated with BLEU, ROUGE, and METEOR. To clarify the objectives of this work, we position **BanglaCHQ-Prantik** as both a research and practical resource. It is designed to support three primary user communities. First, it provides NLP researchers with a benchmark for studying low-resource dialectal translation and evaluating prompting strategies in Bangla-family languages. Second, it enables machine translation developers to fine-tune and assess large language models or domain-specific translation systems on authentic medical queries. Third, it offers public health com-

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 223–236
December 23, 2025 ©2025 Association for Computational Linguistics

munication practitioners a linguistically grounded tool for producing dialect-sensitive health information and outreach materials. By serving these complementary audiences, **BanglaCHQ-Prantik** bridges the gap between computational research and real-world health communication, reinforcing the dataset's relevance and societal impact.

Our contributions are:

1. We release **BanglaCHQ-Prantik**, a human-validated Sylheti and Chatgaiyya medical translation benchmark with accompanying prompts and scoring scripts.

2. We present the first systematic comparison of open and closed-source LLMs for dialectal medical translation in Bangla, analyzing the effects of prompting strategies.

3. We provide empirical insights into dialectal difficulty, demonstrating that Sylheti is comparatively easier for current large language models (LLMs) than Chittagonian. However, for both dialects, LLM-based translation remains far from accurate or reliable.

2 Dataset Creation

Our aim is to create a new dataset by translating consumer health questions (CHQs) from a Standard Bangla corpus into two major regional dialects: Sylheti and Chittagonian. This process resulted in a parallel corpus where each original data sample is paired with a human-translated version in each dialect. The primary objective is to ensure that the translations are not only linguistically accurate but also preserved all critical clinical information, such as symptoms, durations, and medication details. All human translations have been meticulously reviewed for quality, focusing on accurate terminology and natural phrasing.

2.1 CHQ Dataset Introduction

Our data source is the *BanglaCHQ-Summ* corpus (Khan et al., 2023), which contains original spoken-query passages and their abstractive summaries. This dataset has been chosen because the corpus is situated within the specialized domain of consumer health queries (CHQs). It contains a blend of domain-specific terminology and informal, colloquial expressions, which include conversational elements. By working with this data, we evaluate the models' capacity to not only handle linguistic divergence but also to maintain the fidelity of critical medical information during translation. This presents a more robust and practical challenge than translating standard, well-structured prose.

2.2 Data Filtering and Cleaning

Before starting annotation, we first processed the sourced CHQs to remove irrelevant or inconsistent text while keeping all clinical content intact. **Text Cleaning and Normalization.** We corrected obvious formatting issues that break downstream processing such as: stray characters (i.e. isolated '#' tokens), repeated whitespace, inconsistent punctuation and standardized numerals and measurement units to a consistent textual form (e.g., '5 mg', '5mg' → '5 mg'). We avoided any semantic rewriting. Medical terms, dosages, symptom descriptions, and temporal cues are preserved verbatim except for normalization of punctuation/spacing. **Medical Term Validation.** To validate the preprocessing, two native speakers manually inspected a random sample of 600 entries before and after cleaning. They verified that medical entities, dosages, and question intent were preserved. No loss of clinical information was observed. In these spot checks, we also observed that cleaning consistently fixed tokenization and formatting errors while preserving clinical content and the original question intent/answerability.

2.3 Data Annotation

Nine native speakers produced the reference translations: six for Sylheti and three for Chittagonian. All were native speakers from diverse backgrounds (high school, university graduates, and working adults). Before beginning, annotators received written guidelines with example translations based on four principles: (i) preserve medical meaning, (ii) use idiomatic dialect phrasing, (iii) apply consistent orthography, and (iv) retain terminology, numerals, and units.

The corpus was partitioned by dialect and stratified by length and topic. The 2,350 Sylheti items were split roughly evenly among six annotators, while 500 Chittagonian items were divided among three. Two medical experts independently reviewed the dataset to validate the accuracy of clinical terminology. Translation took 11 days. Annotators gave informed consent, submitted anonymized translations, and were paid Tk. 2 per item. Annotators self-reviewed their work and the study team ran random spot checks, returning flagged items for

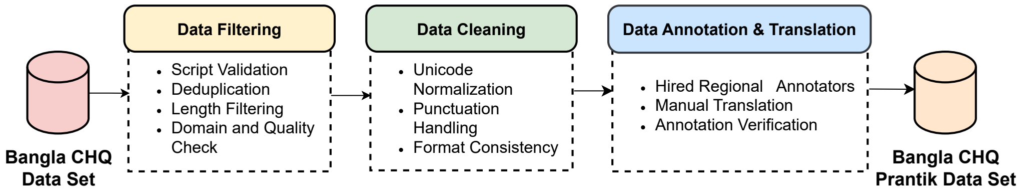


Figure 1: Data processing pipeline for constructing the Bangla CHQ Prantik dataset. The raw Bangla CHQ data undergoes data filtering (script validation, deduplication, length filtering, domain checks), followed by data cleaning (Unicode normalization, punctuation handling, format consistency), and finally data annotation and translation by regional annotators before producing the finalized dataset.

minor corrections.

2.4 Data Validation

We employed eleven independent validators to ensure the quality and accuracy of our translations. Given that our parent dataset BangCHQ already contained validated medical terminology, our emphasis was on recruiting validators with diverse linguistic backgrounds who could assess the naturalness and appropriateness of regional language use, rather than primarily focusing on medical expertise.

Sylheti. The 2,350 items were divided into two equal halves. Validators A and B jointly reviewed the first 1,175 items, while Validators C and D reviewed the second half. A fifth validator (E) served as adjudicator for any disagreements. Inter-validator agreement was assessed using BLEU, BERTScore, METEOR, and ROUGE-L (F1) metrics. For Validators A & B, the scores were 73.28%, 95.24%, 82.77%, and 86.43%, respectively; for C & D, the scores were 71.12%, 93.49%, 80.26%, and 82.57%.

Chittagonian. Validators E and F independently reviewed all 500 items. A third validator adjudicated disagreements, with the majority vote determining the final version. Agreement was again measured using BLEU, BERTScore, METEOR, and ROUGE-L (F1), yielding scores of 70.97%, 94.30%, 79.94%, and 81.18%, respectively.

Expert Medical Validation. To ensure clinical accuracy, we conducted an additional expert validation phase with two licensed physicians who independently reviewed a random sample of 75 items from the combined dataset. The medical experts assessed the preservation of clinical intent and the appropriateness of translated medical terminology in regional contexts. Inter-annotator agreement between the two physicians, measured using BLEU, BERTScore, METEOR, and ROUGE-L (F1), yielded scores of 89.45%, 97.12%, 91.33%, and 93.76%, respectively, indicating very high consistency in their clinical assessments.

Validators reviewed spelling, numerals, clinical terminology, and fidelity to the original question's intent. While automatic similarity metrics offered rough signals of agreement, all final decisions were human-led. The full validation process took seven days. Appendix tables include flagged items, representative disagreements, and their resolutions.

3 Dataset Statistics

Statistics	Bangla CHQ	Ours	
		Sylheti	Chittagonian
Mean Char. Length	325.71	323.24	310.84
Max Char. Length	868	881	531
Min Char. Length	225	189	181
Mean Word Count	67.50	60.73	57.84
Max Word Count	169	166	93
Min Word Count	39	31	36
#Unique Words	12447	18399	2628
#Unique Sentence	12136	14145	796

Table 1: Dataset statistics of the Bangla CHQ and Sylheti CHQ columns.

The dataset statistics for the proposed **Bangla CHQ Prantik** resource are summarized in Table1. It includes the full set of 2,350 CHQ items translated into Sylheti and a 500-item subset translated into Chittagonian.

The Chittagonian portion is substantially smaller due to practical and linguistic challenges we encountered during data collection. The primary bottleneck was the scarcity of annotators proficient in reading and writing Chittagonian. Unlike Sylheti, which has a more established written tradition and a larger pool of literate speakers, Chittagonian lacks a widely accepted standard orthography and has limited written materials. This has resulted in fewer speakers with the literacy skills needed for specialized translation work, particularly in healthcare. Additionally, the dialect varies considerably across the Chittagong division, requiring annotators to make careful decisions about which forms would be most broadly understood. These factors added substantial time to each translation, as spelling, terminology, and phrasing choices required more deliberation and validation than was typical for Sylheti.

Given these challenges, we prioritized translation accuracy and dialectal authenticity over quantity, recognizing that a smaller, rigorously validated dataset would be more valuable than a larger corpus of questionable quality. Despite the smaller sample size, the Chittagonian subset still manages to capture essential dialectal variation and serves as an important step toward broader linguistic inclusivity in Bangla health resources. Future work could expand the Chittagonian dataset through several practical steps. Developing clearer orthographic guidelines in collaboration with linguistic experts would give annotators a more consistent framework to work from. A phased expansion approach could also prove effective, where new translations go through community review to validate quality while simultaneously training additional annotators. Machine translation may eventually help with initial drafts, though given the dialect's complexity, human oversight will remain essential for the foreseeable future.

4 Experiment Setup

Zero-Shot Prompting. To evaluate the capabilities of LLMs, we first employ the *zero-shot prompting* approach. Each model is provided with a system prompt P and a dialect text T_{Dialect}, and is tasked with generating the corresponding formal text T_{Formal} without access to any example pairs (Brown et al., 2020).

Few-Shot Prompting. In the *few-shot prompting* setting, in addition to the system prompt P and input dialect text T_{Dialect}, the model is also given a set of k labeled example pairs $\mathbf{E} = (T_D^1, T_F^1), \ldots, (T_D^k, T_F^k)$, where each (T_D^i, T_F^i) pair represents a dialect sentence and its corresponding formal version (Brown et al., 2020).

Chain-of-Thought (CoT) Prompting. For *Chain-of-Thought (CoT)* prompting, we guide the model to engage in intermediate reasoning before generating a response (Wei et al., 2022). This is achieved by appending the phrase *"Let's think step by step"* to the system prompt P, encouraging the model to produce multi-step inferences leading up to the final output (Kojima et al., 2022). In our translation setting, we further instructed the model to analyze the text step by step before translating, prompting it to reason about meaning, structure, and terminology prior to producing the final translation. Details of the prompt configurations are provided in the Appendix D.

5 Result and Analysis

Model Performance. Closed-source systems (GPT-4o, Gemini 2.5) outperform open-source baselines (Qwen 2.5 3B, Gemma 3 1B), reflecting advantages in scale, alignment, and multilingual corpora. Gemini 2.5 is strongest especially on Sylheti with 5-shot prompting (BLEU 23.67, ROUGE-1 49.02, METEOR 43.82) likely due to broader linguistic coverage and more effective Bangla tokenization. GPT-4o is competitive but slightly lower; Gemma 3 1B surpasses Qwen 2.5 3B, while both open models show limited exposure to dialectal Bangla.

Prompting Impact. Few-shot prompting yields the most reliable gains, notably for Gemini 2.5, by supplying lexical/morphological anchors for medical phrasing. Chain-of-thought (CoT) shows smaller, less stable gains, sometimes falling below both 1-shot and 5-shot, likely because explicit reasoning overgeneralizes Sylheti and Chittagonian's repetitive syllables and reduces lexical fidelity.

Dialect-wise Analysis. Sylheti consistently outperforms Chittagonian across models. Even with CoT, Chittagonian lags (BLEU 21.53 *vs.* Sylheti ROUGE-1 33.37), showing greater differences in pronunciation and structure compared to Standard Bangla, sparser pretraining exposure, and less standardized orthography. Sylheti benefits from wider representation in online and digital resources, contributing to more stable spelling conventions.

Error Analysis. The main errors are (i) terminology mismatches or omissions, (ii) numeral/unit mistakes, (iii) literalized idioms, and (iv) orthographic drift. Closed-source models better preserve medical terms, whereas open models tend to translate too literally, often missing contextual nuances and producing less adequate outputs. Few-shot examples reduce numeric and lexical issues but leave

Model Name	**Sylheti**					**Chittagonian**				
	B	**R1**	**R2**	**R_L**	**Met**	**B**	**R1**	**R2**	**R_L**	**Met**
					Zero Shot Prompt					
Open Source										
Qwen 2.5 3B	9.4	32.97	14.76	31.18	22.47	3.35	16.83	4.2	15.71	10.15
Gemma 3 1B	12.64	33.84	12.41	32.13	25.84	6.93	24.04	6.99	23.3	18.44
Closed Source										
GPT-4o	22.9	46.36	22.98	45.61	41.55	9.08	27.74	8.76	27.46	22.86
Gemini 2.5	20.97	46.33	22.56	45.4	41.09	14.66	37.12	15.61	37.06	32.86
					1 Shot Prompt					
Open Source										
Qwen 2.5 3B	8.68	29.10	13.01	27.50	19.99	3.12	15.43	3.75	14.53	9.27
Gemma 3 1B	13.52	38.17	15.87	35.18	33.29	5.05	19.89	5.67	18.32	16.43
Closed Source										
GPT-4o	22.23	45.95	22.32	45.19	40.92	9.02	28.04	8.94	27.81	23.02
Gemini 2.5	22.19	47.49	23.82	46.64	42.34	11.56	33.61	12.85	33.41	28.99
					5 Shot Prompt					
Open Source										
Qwen 2.5 3B	8.57	31.38	13.61	29.28	21.11	3.36	16.35	4.06	15.21	9.91
Gemma 3 1B	17.24	40.29	17.48	38.78	33.44	7.22	23.97	7.21	23.51	19.1
Closed Source										
GPT-4o	22.34	46.28	22.42	45.38	40.78	8.99	29.07	9.11	28.67	23.44
Gemini 2.5	23.67	49.02	25.2	49.97	43.82	14.78	36.76	14.95	38.85	34.96
					CoT Prompt					
Open Source										
Qwen 2.5 3B	7.16	29.41	11.66	26.99	18.93	3.36	16.3	4.06	15.21	9.91
Gemma 3 1B	23.57	47.24	24.88	46.53	43.6	7.34	23.8	6.28	22.74	18.9
Closed Source										
GPT-4o	21.8	45.81	21.6	44.72	39.71	24.67	51.74	26.84	50.57	47.3
Gemini 2.5	21.53	46.84	22.86	46.02	41.54	9.29	33.37	12.1	32.3	28.07

Table 2: Model benchmarking results on the test split of the BANGLA CHQ PRANTIK dataset across four prompting strategies. B, R1, R2, R_L, and Met represent BLEU, ROUGE-1, ROUGE-2, ROUGE-L, and METEOR scores, respectively. Blue text indicates the highest-performing model for each metric within each prompting strategy configuration (Zero Shot, 1 Shot, 5 Shot, and CoT).

idiomatic and orthographic inconsistencies largely unresolved (see Appendix C).

6 Conclusion

We present **BANGLA CHQ PRANTIK** , extending BanglaCHQ-Summ with human-validated translations for two major Bangla dialects—*Sylheti* and *Chittagonian*. Across zero-shot, few-shot, and CoT settings, five LLMs were evaluated with BLEU, ROUGE, and METEOR; closed-source models (notably Gemini 2.5) consistently outperformed open-source baselines, and Sylheti proved easier than Chittagonian. Persistent errors involved medical terminology, numerals/units, and idioms, underscoring the need for richer dialectal resources. Future work will extend BANGLA CHQ PRANTIK beyond text to *synthetic audio* and *speech-to-text*: developing dialectal TTS for ASR, investigating code-switching and mixed-script cases (Bangla–English, Romanized Bangla), constructing multi-reference test sets, and scaling human evaluation of adequacy, fluency, and dialect authenticity (Khan et al., 2023). For healthcare applications, priorities include robustness to spoken input and user-centered evaluation to ensure accuracy, safety, and usability.

Limitations

Our work has several limitations. First, evaluation relies on single-reference translations, which penalize legitimate dialectal variation in spelling or phrasing. Second, closed-source systems are black-boxes, limiting insight into their training data or dialectal exposure. Third, automatic metrics (BLEU, ROUGE, METEOR) cannot fully capture the clinical adequacy of translations; a human adequacy/fluency study would strengthen conclusions. Finally, we focus exclusively on text translation; extending to spoken dialects and multimodal contexts is an important future step.

Ethics Statement

BanglaCHQ-Prantik is developed entirely from anonymized consumer health queries sourced from the publicly available BanglaCHQ-Summ corpus. The dataset contains no personally identifiable information, and no such information was collected, stored, or distributed at any stage of the project. All dialectal translations were produced by native speakers who provided informed consent and received fair compensation for their work; no sensitive personal or demographic details about annotators are included. The dataset focuses solely on linguistic variation within medical-domain text and does not contain material intended to harm, stigmatize, or misrepresent any individual or community. Although the corpus involves health-related content, it is not designed or intended for clinical decision-making or the provision of medical advice. Based on these considerations, we do not anticipate any ethical concerns associated with the release or use of BanglaCHQ-Prantik.

Acknowledgements

Our work is supported by the Islamic University of Technology Research Seed Grants (IUT RSG) (Ref: REASP/IUT-RSG/2022/OL/07/012).

References

2025. Chittagonian (chit1275) — glottolog 5.2. https://glottolog.org/resource/languoid/id/chit1275. Accessed: 2025-10-04.

2025. Sylheti (sylh1242) — glottolog 5.2. https://glottolog.org/resource/languoid/id/sylh1242. Accessed: 2025-10-04.

Hilal Al Shamsi, Abdullah G. Almutairi, Sulaiman Al Mashrafi, and Talib Al Kalbani. 2020. Implications of language barriers for healthcare: A systematic review. *Oman Medical Journal*, 35(2):e122.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization (MTEval 2005)*, pages 65–72.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sinthia Chowdhury, Deawan Rakin Ahamed Remal, Sheak Rashed Haider Noori, Syed Tangim Pasha, and Ashraful Islam. 2025. Chatgaiyyaalap: A dataset for conversion from chittagonian dialect to standard bangla. *Data in Brief*, 59:111413. Dataset version: Mendeley Data V1, DOI:10.17632/wtms9xbkkw.1.

Directorate General of Health Services (DGHS). 2021. Covid-19 (bangladesh covid-19 management guideline). https://dghs.gov.bd/site/page/86973b0c-62dd-4a43-92f2-b305e3264c88/. Official public health guideline published in Standard Bangla by the Directorate General of Health Services, Ministry of Health and Family Welfare, Bangladesh.

Directorate General of Health Services (DGHS). 2022. Mental health and covid-19 guideline. https://dghs.gov.bd/site/page/86973b0c-62dd-4a43-92f2-b305e3264c88/. Standard Bangla document under the National Health Communication Program, Bangladesh.

Alvi Khan, Fida Kamal, Mohammad Abrar Chowdhury, Tasnim Ahmed, Md Tahmid Rahman Laskar, and Sabbir Ahmed. 2023. BanglaCHQ-summ: An abstractive summarization dataset for medical queries in Bangla conversational speech. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 85–93, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Ministry of Health and Family Welfare (MOHFW). 2024. Ministry of health and family welfare, government of bangladesh: Official portal (in standard bangla). https://mohfw.gov.bd/. Official Bangla-language public health resources, circulars, and program descriptions.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation (WMT): Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics. Introduces sacreBLEU for standardized BLEU reporting.

Nusrat Sultana, Rumana Yasmin, Bijon Mallik, and Mohammad Shorif Uddin. 2025. Onubad: A comprehensive dataset for automated conversion of bangla regional dialects into standard bengali dialect. *Data in Brief*, 58:111276. Dataset article; freely accessible via Mendeley at https://data.mendeley.com/datasets/6ft99kf89b/2.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

A Dataset Comparison

We showed a comparison between different dialects of a single text in our Bangla CHQ Prantik dataset. The comparison is shown in the Figure 2, with English translation provided in the diagram to aid comprehension. This figure illustrates dialectal variation in medical-domain text, showing how a single health-related message in English is translated into Standard Bangla, Chittagonian, and Sylheti. While the core meaning is preserved across versions, there are notable differences in vocabulary, morphology, and phrasing. For instance, the word for "ear" appears in the three variants, and verbs such as "hurt" and "came out" are expressed using distinct regional forms.

These differences highlight the linguistic distance between Standard Bangla and regional dialects, which often lack standardized orthographies and are not mutually intelligible. Such variation poses challenges for both human understanding and machine translation in healthcare settings. This example underscores the importance of dialect-specific resources—such as BANGLA CHQ PRANTIK ensure equitable access to medical information for all language communities in Bangladesh.

B Experiment Details

Gemma Settings: Experiments on Gemma were conducted on Kaggle Notebooks with Python 3.11.13. Hardware resources consisted of two NVIDIA Tesla T4 GPUs (16 GB VRAM each), 4 vCPUs, and 31 GB system RAM. The model was executed with `float32` precision and GPU memory utilization set to 0.7, using tensor parallelism across two GPUs. The maximum model length was configured to 2048 tokens, with up to 24 parallel sequences.

Qwen Settings: Experiments on Qwen2.5 3B were conducted on Kaggle Notebooks with Python 3.11.13. Hardware resources included an NVIDIA Tesla T4 GPU (16 GB VRAM) alongside 4 vCPUs and 31 GB of system RAM. The model was run with `float16` precision and GPU memory utilization set to 0.65. The maximum model length was configured to 2048 tokens, with up to 24 parallel sequences.

Gemini: For Gemini, the gemini-2.5-flash model was accessed through the official API. All inference requests were handled remotely, following the platform's default settings.

GPT: Experiments with GPT used the GPT-4o-mini model via the official OpenAI API. Inference was performed with default API settings.

B.1 Evaluation Metrics

We evaluated system outputs against human references using three standard MT families: **BLEU**, **ROUGE** (1/2/L), and **METEOR**. For each translation pair we compute the metrics below, then report corpus-level statistics (mean, median, min/max, std) for the main dataset. Results are shown separately for Sylheti and Chittagonian under 0-shot, 5-shot, and CoT settings.

BLEU (higher is better): We report corpus-level BLEU using **sacreBLEU** (Post, 2018), which provides a standardized evaluation pipeline and generates a unique signature to ensure exact reproducibility. For Bengali-script text, we use sacreBLEU's built-in tokenization (with no custom preprocessing) to ensure consistency. Final scores are averaged at the corpus level and presented as percentages in column **B**.

ROUGE-1/2/L (higher is better): We compute ROUGE-1 and ROUGE-2 (unigram/bigram overlap) and ROUGE-L (longest common subsequence) using the official `rouge_score` implementation (Lin, 2004). We report the F_1 variant without stemming, again as percentages (columns **R1**, **R2**, **R_L**).

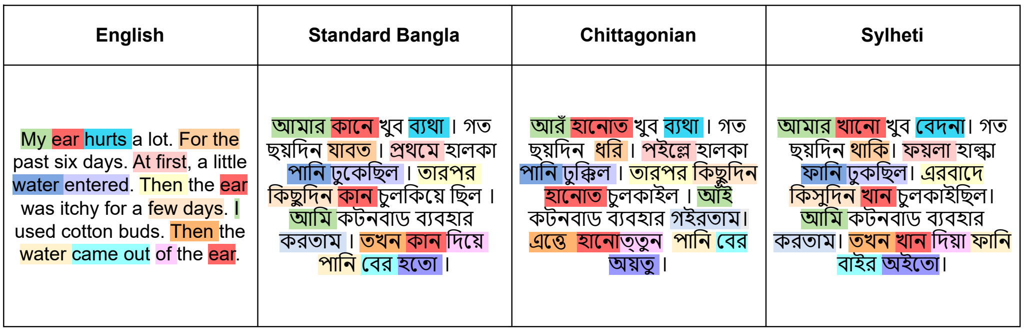
METEOR (higher is better): METEOR (Banerjee and Lavie, 2005) is computed with NLTK's implementation over whitespace tokens (no stemming or paraphrase tables), yielding a precision/recall–balanced score. We report percentages in the **Met** column. METEOR rewards partial matches and recall, making it more sensitive to meaning preservation and synonymy—particularly valuable for dialectal Bangla, where lexical variation and spelling differences are common.

Interpretation: BLEU emphasizes n-gram precision with a brevity penalty; ROUGE captures lexical overlap and sequence alignment; METEOR balances precision and recall with alignment heuristics. These metrics complement each other by reflecting both word-level accuracy and sequence-level similarity in dialectal translations.

C Extended Error Analysis and Findings

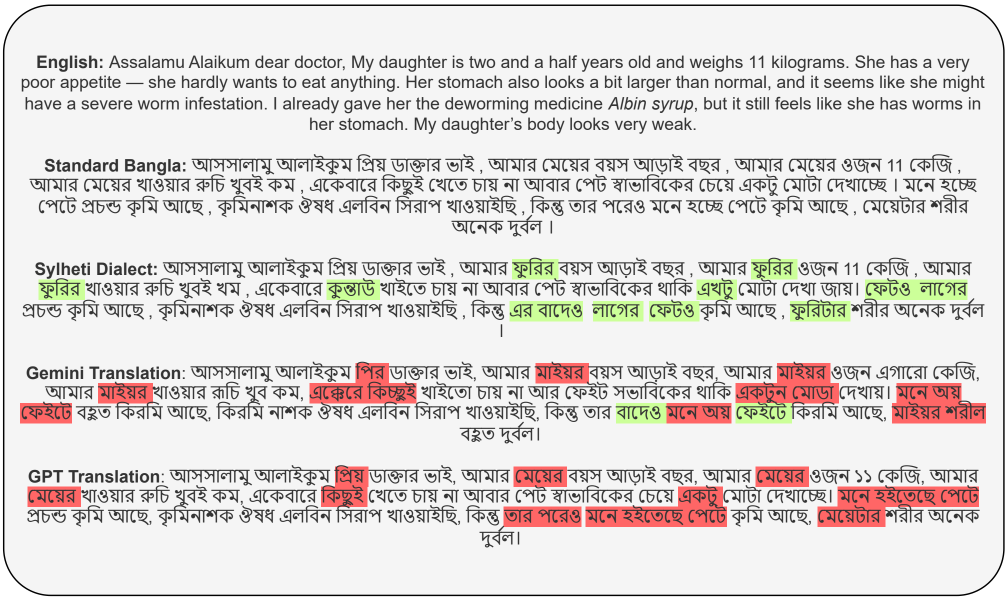
C.1 Model Performance

Closed-source models (GPT-4o, Gemini 2.5) consistently outperform open-source ones across all settings. Gemini 2.5 shows particular robustness in Sylheti, leveraging in-context learning to achieve state-of-the-art results. GPT-4o remains strong across both dialects, though generally a step behind. Among open-source models, Gemma 3 1B consistently outperforms Qwen 2.5 3B, reflecting stronger instruction tuning. However, both open



English	Standard Bangla	Chittagonian	Sylheti
My ear hurts a lot. For the past six days. At first, a little water entered. Then the ear was itchy for a few days. I used cotton buds. Then the water came out of the ear.	আমার কানে খুব ব্যথা । গত ছয়দিন যাবত । প্রথমে হালকা পানি ঢুকেছিল । তারপর কিছুদিন কান চুলকিয়ে ছিল । আমি কটনবাড ব্যবহার করতাম । তখন কান দিয়ে পানি বের হতো ।	আরঁ হানোত খুব ব্যথা । গত ছয়দিন ধরি । পইল্লে হালকা পানি ঢুক্কিল । তারপর কিছুদিন হানোত চুলকাইল । আঁই কটনবাড ব্যবহার গইরতাম। এত্তে হানোত্তুন পানি বের অয়তু ।	আমার খানো খুব বেদনা। গত ছয়দিন থাকি। ফয়লা হাল্কা ফানি ঢুকছিল। এরবাদে কিসুদিন খান চুলকাইছিল। আমি কটনবাড ব্যবহার করতাম। তখন খান দিয়া ফানি বাইর অইতো।

Figure 2: Demonstration of dialectal variation in medical-domain text. A Standard Bangla sentence (left) is shown alongside its Chittagonian (middle) and Sylheti (right) translations. Highlighted segments illustrate differences in word choice, morphology, and phrasing across dialects.



English: Assalamu Alaikum dear doctor, My daughter is two and a half years old and weighs 11 kilograms. She has a very poor appetite — she hardly wants to eat anything. Her stomach also looks a bit larger than normal, and it seems like she might have a severe worm infestation. I already gave her the deworming medicine *Albin syrup*, but it still feels like she has worms in her stomach. My daughter's body looks very weak.

Standard Bangla: আসসালামু আলাইকুম প্রিয় ডাক্তার ভাই , আমার মেয়ের বয়স আড়াই বছর , আমার মেয়ের ওজন 11 কেজি , আমার মেয়ের খাওয়ার রুচি খুবই কম , একেবারে কিছুই খেতে চায় না আবার পেট স্বাভাবিকের চেয়ে একটু মোটা দেখাচ্ছে । মনে হচ্ছে পেটে প্রচন্ড কৃমি আছে , কৃমিনাশক ঔষধ এলবিন সিরাপ খাওয়াইছি , কিন্তু তার পরেও মনে হচ্ছে পেটে কৃমি আছে , মেয়েটার শরীর অনেক দুর্বল ।

Sylheti Dialect: আসসালামু আলাইকুম প্রিয় ডাক্তার ভাই , আমার ফুরির বয়স আড়াই বছর , আমার ফুরির ওজন 11 কেজি , আমার ফুরির খাওয়ার রুচি খুবই খম , একেবারে কুন্তাউ খাইতে চায় না আবার পেট স্বাভাবিকের থাকি এখটু মোটা দেখা জায়। ফেটও লাগের প্রচন্ড কৃমি আছে , কৃমিনাশক ঔষধ এলবিন সিরাপ খাওয়াইছি , কিন্তু এর বাদেও লাগের ফেটও কৃমি আছে , ফুরিটার শরীর অনেক দুর্বল ।

Gemini Translation: আসসালামু আলাইকুম পির ডাক্তার ভাই, আমার মাইয়র বয়স আড়াই বছর, আমার মাইয়র ওজন এগারো কেজি, আমার মাইয়র খাওয়ার রূচি খুব কম, এক্কেরে কিচ্ছুই খাইতো চায় না আর ফেইট সভাবিকের থাকি একটুন মোডা দেখায়। মনে অয় ফেইটে বহুত কিরমি আছে, কিরমি নাশক ঔষধ এলবিন সিরাপ খাওয়াইছি, কিন্তু তার বাদেও মনে অয় ফেইটে কিরমি আছে, মাইয়র শরীল বহুত দুর্বল।

GPT Translation: আসসালামু আলাইকুম প্রিয় ডাক্তার ভাই, আমার মেয়ের বয়স আড়াই বছর, আমার মেয়ের ওজন ১১ কেজি, আমার মেয়ের খাওয়ার রুচি খুবই কম, একেবারে কিছুই খেতে চায় না আবার পেট স্বাভাবিকের চেয়ে একটু মোটা দেখাচ্ছে। মনে হইতেছে পেটে প্রচন্ড কৃমি আছে, কৃমিনাশক ঔষধ এলবিন সিরাপ খাওয়াইছি, কিন্তু তার পরেও মনে হইতেছে পেটে কৃমি আছে, মেয়েটার শরীর অনেক দুর্বল।

Figure 3: Error analysis across dialectal outputs produced by different models.

models struggle with dialectal adaptation, producing literal or fragmented outputs.

C.2 Prompting Impact

Few-shot prompting provides the largest improvements, particularly for Gemini 2.5, which benefits from contextual anchors that guide morphological and lexical choices. CoT prompting, while theoretically promising, offers only modest improvements and sometimes underperforms compared to 5-shot. This suggests that reasoning-based approaches are less effective for low-resource dialectal translation than concrete in-context examples.

C.3 Dialect-wise Analysis

Sylheti translations consistently achieve higher scores than Chittagonian. For example, even under CoT prompting, Chittagonian lags (BLEU 21.53 vs. Sylheti ROUGE-1 33.37). This disparity reflects two main factors: (i) dialectal distance, as Chittagonian diverges more from Standard Bangla in phonology and morphosyntax; and (ii) data scarcity, since Chittagonian is less likely to appear in large-scale pretraining corpora. Orthographic variability also increases inconsistency, especially in open-source outputs.

C.4 Case Study: Medical Domain Translation Challenges

To illustrate the error patterns observed across models, we present a representative example from the medical domain. The source sentence describes a pediatric case involving deworming medication ("It seems there are worms in the stomach, I gave

Alben syrup deworming medicine, but still it seems there are worms in the stomach, the girl's body is very weak").

The reference Sylheti translation uses distinct dialectal features: *feto* (stomach) instead of Standard Bangla *pete*, *lager* (seems) for *mone hochhe*, and *furita* (girl) for *meyeta*. Both Gemini and GPT translations exhibit characteristic errors. The Gemini output introduces phonological overcorrections (*feite* instead of *feto*, *kirimi* for *krimi*), demonstrating inconsistent application of dialectal phonology rules. The GPT translation maintains closer orthographic fidelity to Standard Bangla (*pete*, *meyetar*), failing to properly dialectalize key lexical items. Notably, both models preserve the medical term "Alben syrup" correctly, supporting our finding that closed-source models handle domain-specific terminology more reliably. However, dialectal function words show variation: the reference uses *er badeo* (after this), while GPT retains the Standard form *tar poreo*, and Gemini substitutes *tar badeo*—a hybrid form. This example encapsulates the tension between lexical preservation and dialectal authenticity that characterizes medical translation in low-resource settings.

C.5 Error Analysis

We identify four dominant error categories:

1. **Terminology mismatches**: Medical terms (e.g., "antibiotic", "hypertension") are often replaced with generic synonyms or omitted altogether. Closed-source models handle these terms more reliably.

2. **Numerals and units**: Dosages, dates, and measurements are inconsistently translated. Example: "500mg" incorrectly rendered as "5 gram" in Qwen outputs.

3. **Idiomatic expressions**: Dialect-specific idioms are frequently literalized. For instance, Sylheti expressions for pain severity were mapped word-for-word rather than into natural phrasing.

4. **Orthographic drift**: Particularly in Chittagonian, spelling variation leads to inconsistent forms across outputs. This problem is amplified in open-source models.

Few-shot prompting reduces terminology and numeral errors by providing concrete examples. CoT offers some benefit for coverage but does not resolve idiomatic or orthographic inconsistencies.

D Used Prompts in the Paper

Prompt for Bangla to Sylheti Translation (ZeroShot)

ZeroShot Prompt for Bangla to Sylheti Translation

You are a precise translation tool. Your only task is to translate the given Bangla text to Sylheti dialect using Bengali script.

INSTRUCTIONS:

- Translate the Bangla text below to Sylheti dialect
- Use only Bengali script (not Latin script or IPA)
- Return ONLY the translated text with no additional commentary, explanations, or notes
- Do not include phrases like “Here is the translation:” or “The Sylheti translation is:”
- Do not add any metadata, formatting, or extra information
- If you cannot translate a specific word, keep it as is in the original form

Bangla text to translate: `{bangla_text}`

Sylheti translation:

Prompt for Bangla to Sylheti Translation (FewShot)

FewShot Prompt for Bangla to Sylheti Translation

You are a precise translation tool. Your only task is to translate the given Bangla text to Sylheti dialect using Bengali script.

INSTRUCTIONS:

- Translate the Bangla text below to Sylheti dialect
- Use only Bengali script (not Latin script or IPA)
- Return ONLY the translated text with no additional commentary, explanations, or notes
- Do not include phrases like “Here is the translation:” or “The Sylheti translation is:”
- Do not add any metadata, formatting, or extra information
- If you cannot translate a specific word, keep it as is in the original form

Here are some examples of Bangla to Sylheti translations:

Bangla: “Do you meditate regularly?” | Sylheti: [Sylheti translation]
Bangla: “Where do you do coaching?” | Sylheti: [Sylheti translation]
Bangla: “Has the lentil been cooked?” | Sylheti: [Sylheti translation]
Bangla: “Hey there, what are you taking?” | Sylheti: [Sylheti translation]
Bangla: “Will go after prayer” | Sylheti: [Sylheti translation]

Bangla text to translate: `{bangla_text}`

Sylheti translation:

Prompt for Bangla to Sylheti Translation (Chain-of-Thought)

Chain-of-Thought Prompt for Bangla to Sylheti Translation

You are a precise translation tool. Your task is to translate the given Bangla text to Sylheti dialect using Bengali script.

INSTRUCTIONS:

- Analyze the text step-by-step before translating
- Use only Bengali script in your final translation
- After your reasoning, provide ONLY the final translation with no additional commentary

Here are examples showing the translation process:

Ex 1: Bangla: "Do you meditate regularly?" | Reasoning: Phonetic shifts, verb changes | Sylheti: [Translation]

Ex 2: Bangla: "Where do you do coaching?" | Reasoning: Locative form, verb change | Sylheti: [Translation]

Ex 3: Bangla: "Has the lentil been cooked?" | Reasoning: Vowel shift, verb transformation | Sylheti: [Translation]

Ex 4: Bangla: "Hey there, what are you taking?" | Reasoning: Colloquial address, particle transformation | Sylheti: [Translation]

Ex 5: Bangla: "Will go after prayer" | Reasoning: Arabic loanword, future verb ending | Sylheti: [Translation]

Now translate: `{bangla_text}` | **Lets think step by step:** Identify transformations, grammatical changes, phonetic patterns

Provide your final translation below: Sylheti:

Prompt for Bangla to Chittagonian Translation (ZeroShot)

ZeroShot Prompt for Bangla to Chittagonian Translation

You are a precise translation tool. Your only task is to translate the given Bangla text to Chittagonian (Chatgaiyan) dialect using Bengali script.

INSTRUCTIONS:

- Translate the Bangla text below to Chittagonian (Chatgaiyan) dialect
- Use only Bengali script (not Latin script or IPA)
- Return ONLY the translated text with no additional commentary, explanations, or notes
- Do not include phrases like "Here is the translation:" or "The Chittagonian translation is:"
- Do not add any metadata, formatting, or extra information
- If you cannot translate a specific word, keep it as is in the original form

Bangla text to translate: `{bangla_text}`

Chittagonian translation:

Prompt for Bangla to Chittagonian Translation (FewShot)

FewShot Prompt for Bangla to Chittagonian Translation

You are a precise translation tool. Your only task is to translate the given Bangla text to Chittagonian (Chatgaiyan) dialect using Bengali script.

INSTRUCTIONS:

- Translate the Bangla text below to Chittagonian (Chatgaiyan) dialect
- Use only Bengali script (not Latin script or IPA)
- Return ONLY the translated text with no additional commentary, explanations, or notes
- Do not include phrases like “Here is the translation:” or “The Chittagonian translation is:”
- Do not add any metadata, formatting, or extra information
- If you cannot translate a specific word, keep it as is in the original form

Here are some examples of Bangla to Chittagonian translations:

Bangla: “Uncle, will you go to the village house?” | Chittagonian: [Translation]
Bangla: “Do you regularly eat vegetables?” | Chittagonian: [Translation]
Bangla: “Will go after noon prayer” | Chittagonian: [Translation]
Bangla: “Really like spinach” | Chittagonian: [Translation]
Bangla: “What do the maternal grandparents of Rangani do?” | Chittagonian: [Translation]

Bangla text to translate: `{bangla_text}`

Chittagonian translation:

Prompt for Bangla to Chittagonian Translation (Chain-of-Thought)

Chain-of-Thought Prompt for Bangla to Chittagonian Translation

You are a precise translation tool. Your task is to translate the given Bangla text to Chittagonian (Chatgaiyan) dialect using Bengali script.

INSTRUCTIONS:

- Analyze the text step-by-step before translating
- Use only Bengali script in your final translation
- After your reasoning, provide ONLY the final translation with no additional commentary

Here are examples showing the translation process:

Ex 1: Bangla: “Uncle, will you go to the village?” | Reasoning: Vocative particle, possessive/locative | Chittagonian: [Translation]

Ex 2: Bangla: “Do you regularly eat vegetables?” | Reasoning: Pronoun shift, habitual form | Chittagonian: [Translation]

Ex 3: Bangla: “Will go after noon prayer” | Reasoning: Arabic loanword, future verb | Chittagonian: [Translation]

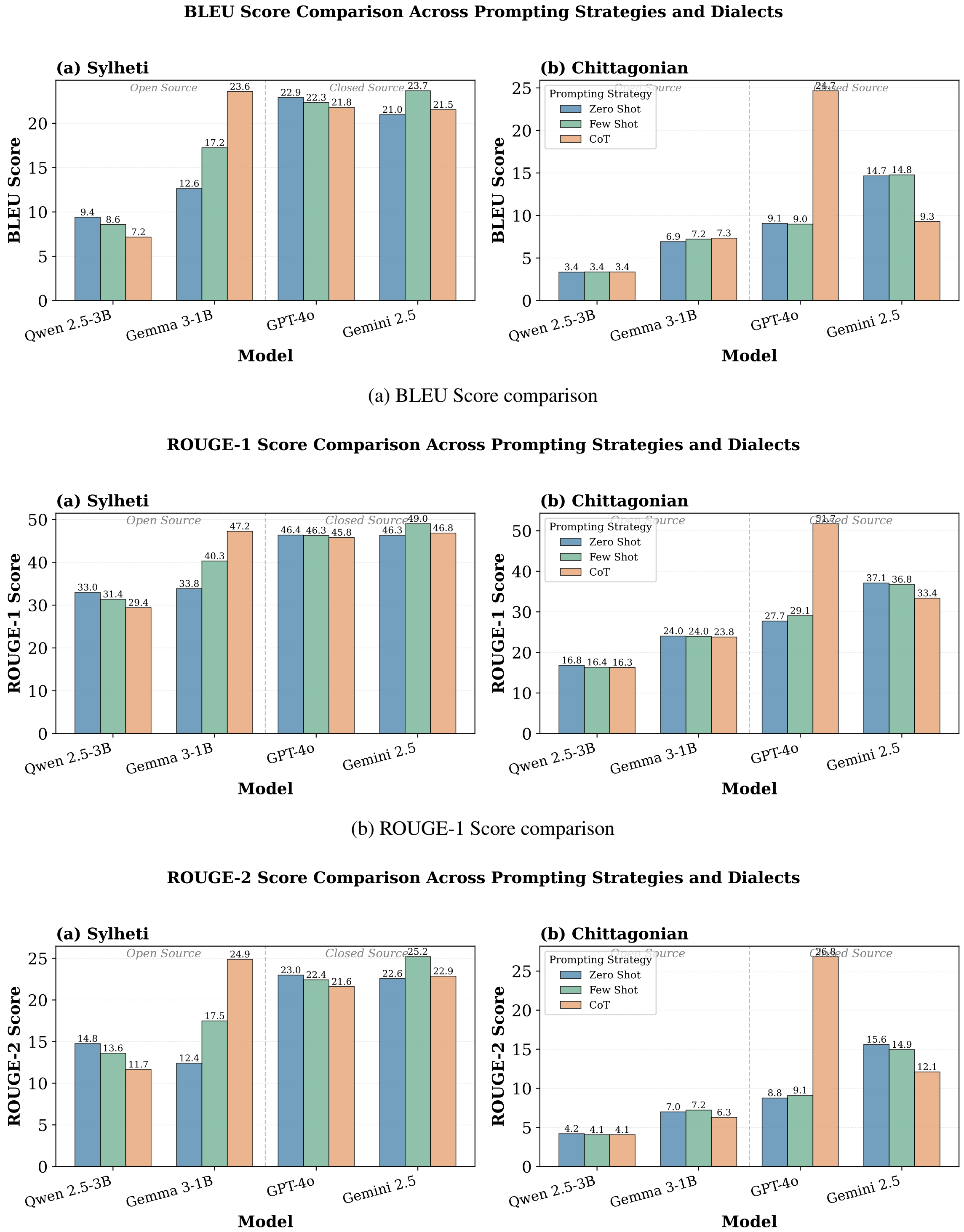
Ex 4: Bangla: “Really like spinach” | Reasoning: Compound unchanged, preference transformation | Chittagonian: [Translation]

Ex 5: Bangla: “What do maternal grandparents do?” | Reasoning: Phonetic compression, plural transformation | Chittagonian: [Translation]

Now translate: `{bangla_text}` | **Let’s think step by step:** Identify transformations, grammatical changes, phonetic patterns

Provide your final translation below: Chittagonian:

E Performance Comparison across Dialects and Prompting Strategies in Different Models

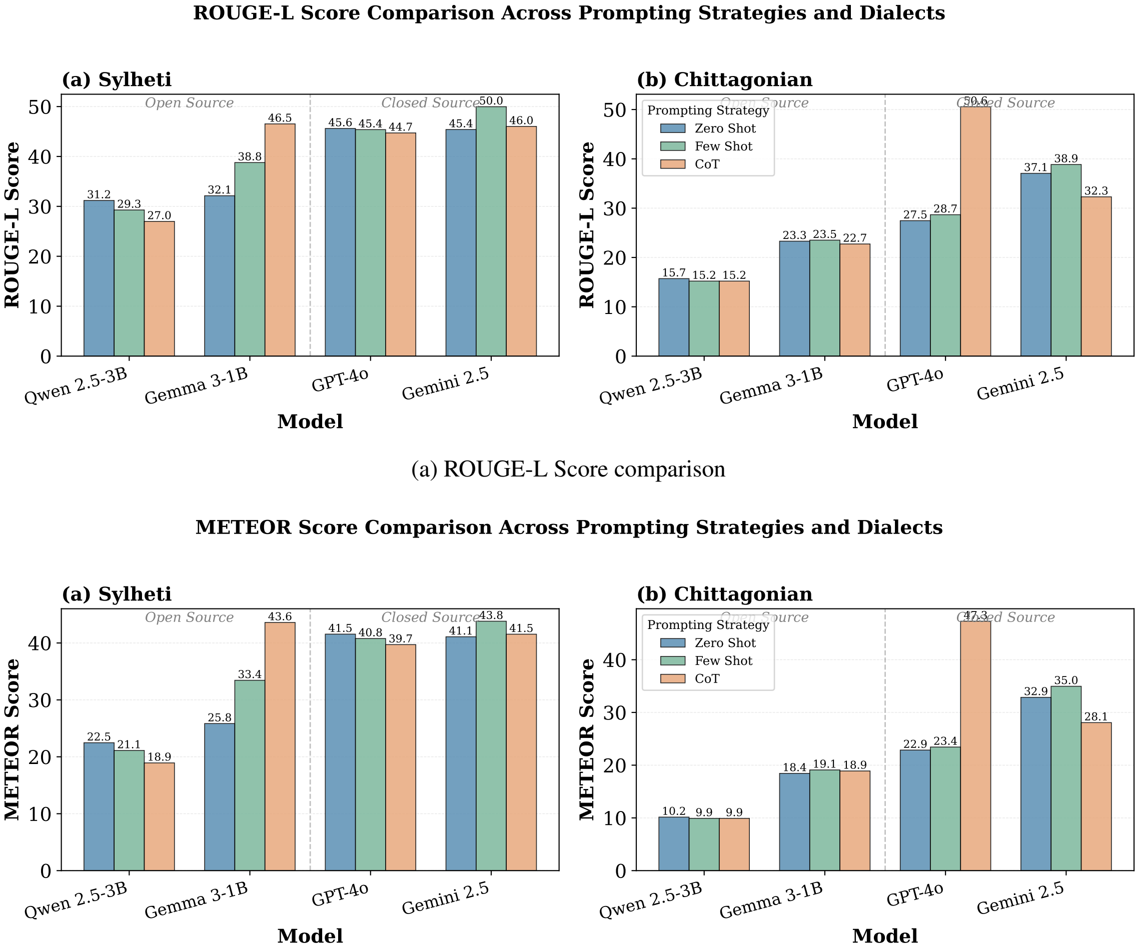


(a) BLEU Score comparison

(b) ROUGE-1 Score comparison

(c) ROUGE-2 Score comparison

Figure 4: Performance comparison across different prompting strategies (Part 1). Comparison of BLEU, ROUGE-1, and ROUGE-2 scores for Sylheti (left) and Chittagonian (right) dialects under Zero Shot, Few Shot, and CoT prompting strategies.



(a) ROUGE-L Score comparison

(b) METEOR Score comparison

Figure 5: Performance comparison across different prompting strategies (Part 2). Comparison of ROUGE-L and METEOR scores for Sylheti (left) and Chittagonian (right) dialects under Zero Shot, Few Shot, and CoT prompting strategies.

BanHate: An Up-to-Date and Fine-Grained Bangla Hate Speech Dataset

Faisal Hossain Raquib[1] *, Akm Moshiur Rahman Mazumder[2] *, Md Tahmid Hasan Fuad[3], Md Farhan Ishmam[4], Md Fahim[2,5]

[1]*Rajshahi University of Engineering and Technology*
[2]*Center for Computational & Data Sciences, IUB*
[3]*University of Manitoba* [4]*University of Utah* [5]*Penta Global Limited*

*Equal Contribution **Correspondence:** {faisal.ece18,fahimcse381}@gmail.com

Abstract

Online safety in low-resource languages relies on effective hate speech detection, yet Bangla remains critically underexplored. Existing resources focus narrowly on binary classification and fail to capture the evolving, implicit nature of online hate. To address this, we introduce BanHate, a large-scale Bangla hate speech dataset, comprising 19,203 YouTube comments collected between April 2024 and June 2025. Each comment is annotated for binary hate labels, seven fine-grained categories, and seven target groups, reflecting diverse forms of abuse in contemporary Bangla discourse. We develop a tailored pipeline for data collection, filtering, and annotation with majority voting to ensure reliability. To benchmark BanHate, we evaluate a diverse set of open- and closed-source large language models under prompting and LoRA fine-tuning. We find that LoRA substantially improves open-source models, while closed-source models, such as GPT-4o and Gemini, achieve strong performance in binary hate classification, but face challenges in detecting implicit and fine-grained hate. BanHate sets a new benchmark for Bangla hate speech research, providing a foundation for safer moderation in low-resource languages. Our dataset is available at: https://huggingface.co/datasets/aplycaebous/BanHate.

Disclaimer: This paper contains potentially offensive content essential to the subject matter.

1 Introduction

Social media has revolutionized human communication, making unprecedented transformations in connecting people and relaying information (Kaplan and Haenlein, 2010). With a user adoption rate of more than 50% and daily spending time exceeding two hours (Gudka et al., 2023), social media has become an integral part of our lives. Yet the evolution of digital technology and online connectivity resulted in the proliferation of online hate content, with studies attesting to the rising trend of hate in mainstream social networks (Goel et al., 2023). Hateful comments can be theorized as means of externalizing internal upheaval and venting bottled-up emotions (The Havok Journal, 2023), with users preferring their native language for emotional salience (Reghunathan and Asha, 2022).

As of 2025, Bangla is spoken natively by over 240 million people (Wikipedia, Ethnologue, 2025), making it one of the most critical languages for online hate moderation. Nevertheless, studies reveal a persistent gap in this domain. While several datasets have been developed for Bengali hate speech detection (Sharif et al., 2022; Romim et al., 2020), they fail to capture the rapidly evolving nature of online discourse, particularly the generational shifts in expressions introduced by Gen Z and Gen Alpha. This necessitates the creation of new, updated Bangla datasets.

Recently, large language models (LLMs) have shown impressive performance in a range of classification tasks, including hate speech detection (Haider et al., 2025). However, the increasing subtlety and implicitness of hateful content continue to pose a challenge. Current models often struggle to infer the underlying intent of such speech and fall short of expectations in detecting implicit hate (Kim et al., 2022).

To bridge this gap, we introduce BanHate, a hate speech dataset constructed from recent YouTube comments. The dataset reflects the linguistic styles of newer generations and captures implicit forms of hate rarely represented in earlier resources. All comments were carefully validated by human annotators to ensure that they contain genuinely targeted speech. Furthermore, we evaluate a suite of LoRA-based fine-tuned models alongside closed-source LLMs on this dataset to assess their effectiveness in identifying implicit hate.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 237–248
December 23, 2025 ©2025 Association for Computational Linguistics

Datasets	#NH	#H	H:NH	Data Source	#Annotators
Sharif et al. (2022)	7,361	8,289	1.12	YouTube, Facebook	2
Belal et al. (2023)	7,585	8,488	1.12	Previous Datasets	2
Romim et al. (2020)	20,000	10,000	0.50	YouTube, Facebook	50
BANHATE (Ours)	10,048	9,155	0.91	Youtube	4

Table 1: **Comparison between existing datasets** based on the number of Non-Hate (#NH), Hate (#H) samples, hate to non-hate ratio (H:NH), source of dataset (Data Source), number of annotators (#Annotators). The hate speech datasets include similar data classes, *e.g.*, aggression and toxic speech.

2 Related Work

Early research on hate speech detection predominantly treated it as a binary classification problem (Djuric et al., 2015; Badjatiya et al., 2017; MacAvaney et al., 2019). Subsequent works expanded to multi-class (Walsh and Greaney, 2025; Hashmi and Yayilgan, 2024) and multi-label formulations (Ilma et al., 2021), enabling more nuanced representations of hate across social, political, and cultural dimensions. Recent efforts have also explored multimodal hate speech detection, integrating textual and visual modalities (Boishakhi et al., 2021; Barua et al., 2024), as well as multilingual and cross-lingual models leveraging architectures such as mBERT to extend detection across languages (Aluru et al., 2020; Ousidhoum et al., 2019).

Despite these advancements, Bangla remains underexplored compared to high-resource languages. Early studies primarily addressed binary hate speech detection (Remon et al., 2022; Das et al., 2022), reflecting the scarcity of linguistic resources. Subsequent research broadened the scope to related domains, including abusive language detection (Aurpa et al., 2022; Emon et al., 2019), cyberbullying (Ahmed et al., 2021; Saifuddin et al., 2023), gender discrimination and sexism (Jahan et al., 2023), and toxic content classification (Belal et al., 2023). More recent efforts have advanced toward multi-label settings, categorizing hate by target domains such as religion, politics, and gender (Sharif et al., 2022; Haider et al., 2025; Romim et al., 2020).

3 The BANHATE Dataset

The creation of BANHATE (**Ban**gla **Hate** Speech Detection) dataset was systematically collected and annotated to address the scarcity of resources for hate speech detection in low-resource languages. It comprises 19,203 curated comments covering seven target groups and seven hate categories, enabling fine-grained analysis (Figure 1).

Video Category	Hate	Non-Hate
Entertainment	2,007	1,762
International	1,720	1,296
News & Politics	3,801	2,541
People & Blogs	585	2,494
Sports	1,042	1,955

Table 3: Hate distribution by video category.

3.1 Data Collection

We collected comments from 328 YouTube videos across five categories: News & Politics, Entertainment, People & Blogs, International, and Sports, spanning April 2024 to June 2025. Only top-level comments were extracted, resulting in 26,730 comments.

3.2 Data Filtering

We removed non-Bangla comments as our dataset's focus is on Bangla hate speech detection, and comments shorter than 20 characters were removed as they lack meaningful context. By filtering, we reduced the dataset to 20,439 comments.

3.3 Data Cleaning

We removed duplicate comments and extraneous content, *e.g.*, URLs, hashtags, and personal identifiers, resulting in 19,203 clean comments for data annotation.

Year	Hate	Non-Hate
2017	105	70
2019	31	364
2020	265	197
2021	58	313
2022	466	1,553
2023	970	1,774
2024	2,780	2,693
2025	4,480	3,084

Table 4: Hate distribution against video upload year.

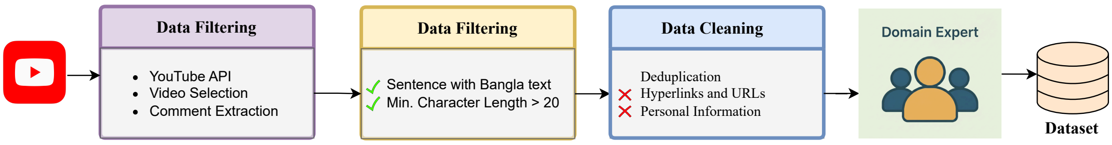


Figure 1: BanHate dataset development pipeline illustrating the four-stage process: (a) data collection from social media platforms, (b) data filtering to remove non-relevant content, (c) data cleaning to eliminate duplicates and extraneous elements, and (d) data annotation & validation.

Hate Category	Entertainment	International	News & Politics	People & Blogs	Sports
Abusive/Violence	571	787	1586	193	174
Body Shaming	94	5	34	15	4
Gender	730	3	201	97	9
Origin	105	304	256	64	46
Personal Offence	1146	662	1942	450	873
Political	46	328	1422	37	168
Religious	98	615	140	47	9

Table 2: Relationship Between hate and video categories.

3.4 Data Annotation

We adopted a two-stage approach for data annotation. First, four native Bangla-speaking undergraduate annotators labeled each comment as Hate or Non-Hate. Their prior experience with social media usage ensured high-quality annotations that captured nuanced hate content. Second, hate comments were further categorized by target group and type of hate. The final labels were determined via majority vote. The annotators were provided monetary compensation. The annotation guidelines provided to the annotators have been reported in Appendix A.

3.5 Data Validation

We evaluated inter-annotator agreement using Cohen's kappa (κ), shown in Table 5. For the primary Hate vs. Non-Hate task, we obtained κ scores of 0.81 (Hate) and 0.75 (Non-Hate), averaging 0.78, which is substantially higher than prior work on content moderation (∼0.53) (Islam et al., 2021), reflecting effective annotator selection and clear guidelines.

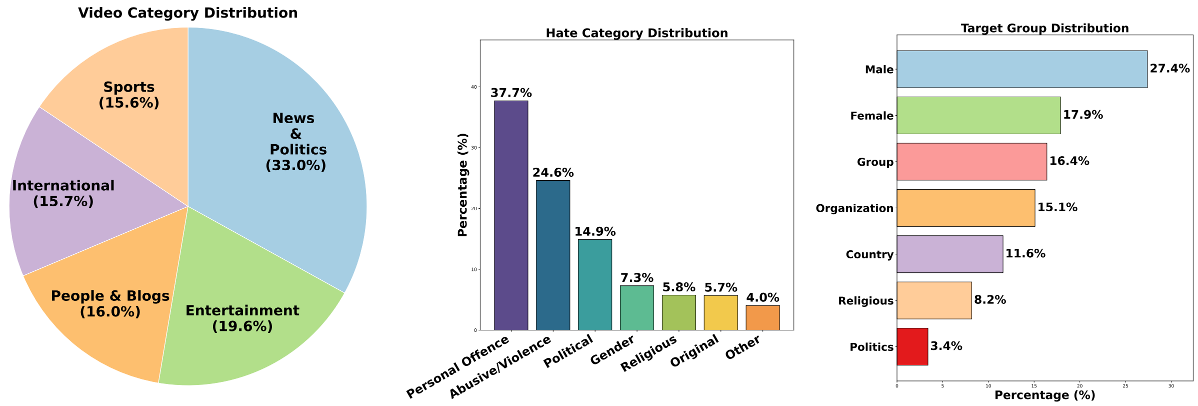
For hate categories, agreement was highest for Personal Offense (0.83) and Abusive/Violent (0.81), where definitions are clearer, and lower for more subjective categories such as Origin (0.67) and Body Shaming (0.69). Among target groups, Female (0.81) and Male (0.77) showed strong agreement, while Group (0.66), Country (0.68), and Religion (0.67) were lower, suggesting that identifying hate toward specific collectives is inherently more challenging and may benefit from additional guidance.

	Label	Kappa(κ)	Avg.
Primary	Hate	0.81	0.78
	Non Hate	0.75	
Hate Categories	Personal Offence	0.83	0.75
	Abusive/Violance	0.81	
	Political	0.74	
	Gender	0.77	
	Religious	0.72	
	Origin	0.67	
	Body Shaming	0.69	
Targeted Groups	Male	0.77	0.72
	Female	0.81	
	Group	0.66	
	Organization	0.69	
	Country	0.68	
	Religion	0.67	
	Politics	0.73	

Table 5: Inter-annotator agreement for the BanHatedataset, measured using Cohen's kappa (κ) across the binary task, hate categories, and targeted groups, with averages indicating substantial reliability.

Type	Total Count	Percentage (%)
Single	5,437	59.39%
Multiple	3,718	40.61%

Table 8: Distribution of single and multiple hate labels in BanHate.



(a) Video Category Distribution

(b) Hate Category Distribution

(c) Target Group Distribution

Figure 2: Distribution of different categories in BANHATE.

Splits		
Train		15362
Test		3841
General Statistics		
Samples		19203
Videos		318
Hate Samples		9,155
Non-Hate Samples		10,048
Video Categories		5
Hate Categories		7
Target Groups		7
Samples	**Hate**	**Non-Hate**
Train	7324	8038
Test	1831	2010
Mean word count	15.78	12.74
Max word count	496	514
Min word count	5	6

Table 6: Dataset statistics of BANHATE.

3.6 Dataset Statistics

Table 6 presents the key statistics of our BANHATE dataset. Table 7 summarizes the major incidents represented in the dataset, spanning July 2024 to June 2025. The most prevalent hate categories observed across these events are Abusive/Violent, Personal Offense, and Political. Figure 2 illustrates the distributions of video categories (Figure 2a), hate categories (Figure 2b), and target groups (Figure 2c). The data reveal that the majority of comments originate from News & Politics videos. Personal Offense and Abusive/Violence account for over 60% of all hate-labeled comments.

4 Experiment Design

We selected a diverse set of LLMs and classified our experimental designs into two categories: (i) Prompt-based Experiments and (ii) LoRA Fine-tuning Experiments.

4.1 Prompt-Based Experiments

For the prompt-based experiments, we considered zero-shot and chain-of-thought prompting (Wei et al., 2022). Details of prompts used in the experimentation are given in the Appendix B.

4.2 LoRA Fine-Tuning

We fine-tuned open-source LLMs using LoRA (Hu et al., 2022), which adapts pre-trained weights efficiently via low-rank updates rather than updating all parameters. Given a pre-trained and frozen weight matrix W, LoRA adds a learnable low-rank update $\Delta W = AB^\top$, where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$ are the trainable matrices, and $r \ll d$. The updated weights are computed as: $W' = W + \Delta W = W + AB^\top$.

This formulation allows the model to capture task-specific knowledge while retaining the generalization capabilities of the pre-trained backbone. The matrices A and B are trained using the loss function defined for the downstream task. In our case, we optimize these parameters using the classification loss $\mathcal{L}_{\text{Class}}$, which denotes the standard cross-entropy loss used for the hate speech classification task. In addition, we introduce a hierarchical loss $\mathcal{L}_{\text{Hier}}$ to jointly model hate category and target group detection, reflecting the dependency between coarse-grained and fine-grained labels. The hierarchical loss is defined as:

$$\mathcal{L}_{\text{Hier}} = \mathcal{L}_{\text{Cat}} + \mathcal{L}_{\text{Group|Cat}},$$

Time Period	Event	Samples	Major Hate Categories
Apr’ 2025	Pahalgam attack	546	Origin, Religious, Personal
Jun’ 2025	Iran - Israel War	1638	Abusive/Violence, Religious, Personal
May’ 2025	Ind-Pak War	707	Abusive/Violence, Personal
July’24 - Aug’24	Quota Reform Movement	1542	Abusive/Violence, Political, Personal
Aug’24 - Nov’24	Post-Regime Change Events	1483	Abusive/Violence, Political, Personal

Table 7: Events Covered in Dataset with Number of Hate Samples and Major Hate Categories

Target Group	A/V	Body Shaming	Gender	Origin	P Off	Political	Religious
Country	587	0	0	583	444	317	147
Female	528	90	947	39	1264	132	71
Group	874	18	75	159	861	269	67
Male	1034	54	112	98	2293	457	115
Organization	632	0	119	75	824	899	33
Politics	182	0	11	21	79	240	24
Religious	310	1	21	100	209	225	792

Table 9: Target Group & Hate Category Relation. A/V refers to Abusive/Violence and P Off refers to Personal Offence respectively.

where $\mathcal{L}_{\text{Cat}}$ is the cross-entropy loss for hate category prediction, $\mathcal{L}_{\text{Group|Cat}}$ is the conditional loss for target group prediction given the predicted category. We configured LoRA with $\alpha = 64$, $r = 64$, a dropout rate of 0.01, a learning rate of 1×10^{-4}, and a batch size of 4 to 32. LoRA was applied to all weight matrices of the pre-trained models, and each model was fine-tuned for a single epoch. For inference, we used VLLM (Kwon et al., 2023), while LLaMA-Factory (Zheng et al., 2024) was used for LoRA fine-tuning. To ensure reproducibility, an evaluation is performed using greedy decoding with a temperature of 0 and no sampling.

4.3 Baselines

We evaluate five open-source models: Qwen-2.5-7B (Qwen et al., 2025), Gemma-3-12B (Team et al., 2025), Llama-3.1-8B (Dubey et al., 2024), Phi-4 14B (Abdin et al., 2024), and Mistral 7B (Jiang et al., 2023), and two closed-source models: Gemini 2.5 Flash and GPT 4o.

5 Result and Analysis

5.1 Hate Speech Detection

Table 12 compares five open-source LLMs on Precision, Recall, and F1 across three setups: Zero-Shot, Chain-of-Thought (CoT), and LoRA Fine-Tuning.

5.1.1 Zero-Shot Prompting

Gemma-3 12B leads with the highest F1 for Non-Hate (84.76%) and Hate (80.64%), showing strong generalization capabilities. Mistral 7B achieved high Precision (Hate 97.08%) and Non-Hate Recall (97.62%) but low Hate Recall (60.15%), resulting in a moderate F1 (74.28). Phi-4 had the highest Hate Recall (82.58%) but extremely low Precision, yielding an F1 of 18.27%. LLaMA-3.1 8B and Qwen-2.5 7B showed average performance, with Qwen favoring recall over precision.

5.1.2 Chain-of-Thought (CoT) Prompting

Gemma-3 12B performed consistently, reflecting its robustness in the CoT setting. LLaMA-3.1 8B improved notably compared to zero-shot prompting, reaching Non-Hate F1 of 80.94%, highlighting substantial gains from explicit reasoning. Mistral showed an imbalance: high non-hate precision (93.91%) and hate recall (98.14%), but non-hate recall fell to 26.17%, thus, reducing hate F1 to 40.94%. Qwen-2.5 7B also degraded performance with hate F1 dropping to 33.99%.

5.1.3 LoRA Fine-Tuning.

LoRA fine-tuning yields the most consistent and substantial performance gains across all models. LLaMA-3.1 8B achieves the highest F1 scores for both Non-Hate (85.96%) and Hate (83.83%) classes, demonstrating effective task adaptation.

Models	Hate Category (F1)							Overall Metrics			
	P Off	A/V	Pol	Gen	Rel	Ori	BS	Micro	Macro	Subset Acc	Hamming
				Zero Shot Prompting							
Qwen-2.5-7B	17.28	28.47	41.10	22.22	49.66	21.65	16.13	27.43	28.07	14.25	21.13
Gemma-3-12B	**66.43**	**50.05**	44.41	39.04	**64.29**	25.45	08.94	**51.25**	**42.66**	**27.25**	20.29
Llama-3.1-8B	61.36	40.06	42.54	**46.94**	40.82	04.85	**33.47**	17.40	15.96	02.08	**40.29**
Phi-4 14B	54.69	37.49	**49.78**	17.19	56.57	**28.24**	08.89	43.92	36.12	21.57	22.81
Mistral 7B	34.01	02.59	36.91	18.57	24.88	15.06	06.56	23.28	19.80	01.80	39.13
				Chain of Thought (CoT)							
Qwen-2.5-7B	18.18	29.06	10.55	**29.83**	53.38	12.97	04.60	23.35	22.65	09.63	19.35
Gemma-3-12B	52.79	**58.78**	**48.57**	28.88	**61.27**	21.00	**17.50**	**51.46**	**41.26**	**14.34**	19.17
Llama-3.1-8B	53.45	51.19	34.42	10.53	54.86	**34.06**	06.56	45.69	35.01	10.84	19.17
Phi-4 14B	15.59	44.33	21.58	27.14	57.74	25.81	13.04	29.72	29.32	10.07	19.60
Mistral 7B	**67.26**	48.45	11.79	03.69	19.47	17.17	08.89	44.10	25.25	01.42	**30.88**
				LoRA Fine-Tuning							
Qwen-2.5-7B	59.93	52.58	48.91	54.34	68.41	35.51	00.00	55.03	45.67	63.11	08.12
Gemma-3-12B	**63.81**	**55.36**	53.60	**56.78**	**71.33**	40.01	13.63	**58.58**	50.65	**64.79**	10.23
Llama-3.1-8B	63.68	54.14	**54.29**	55.21	70.24	**40.51**	**23.26**	58.12	**51.62**	62.46	08.34
Phi-4 14B	60.74	51.47	51.13	47.35	67.21	36.36	15.00	55.00	47.04	60.46	08.56
Mistral 7B	60.97	51.71	50.46	53.20	67.89	35.21	11.76	55.55	47.31	42.30	**12.81**

Table 10: Performance Analysis of the models on Hate Category. P Off, A/V, Pol, Gen, Rel, Ori, and BS refer to Personal Offence, Abusive/Violence, Political, Gender, Religious, Origin, and Body Shaming, respectively. Color marks the highest performance for each configuration and metric.

Models	Target Group							Overall			
	Male	Female	Group	Org	Country	Rel	Pol	Micro	Macro	Subset Acc	Hamming
				Zero Shot Prompting							
Qwen-2.5-7B	06.92	18.47	10.55	21.33	35.97	26.67	**33.87**	19.65	21.97	**11.08**	18.49
Gemma-3-12B	**54.71**	67.84	**34.25**	34.17	**50.25**	**43.68**	06.83	**38.66**	**41.68**	00.54	38.02
Llama-3.1-8B	46.46	54.70	30.28	35.05	40.96	39.16	11.98	36.76	36.94	00.81	38.22
Phi-4 14B	32.69	**67.90**	31.38	**40.45**	39.77	31.86	11.31	35.86	36.48	04.97	32.76
Mistral 7B	00.00	20.41	32.47	04.96	38.78	26.73	10.84	23.38	19.17	00.93	**38.71**
				Chain of Thought							
Qwen-2.5-7B	17.89	35.41	29.12	06.81	37.88	39.39	04.65	26.89	24.45	15.59	18.41
Gemma-3-12B	06.18	**57.04**	32.53	15.56	33.70	41.35	09.20	28.75	27.94	03.07	27.41
Llama-3.1-8B	**48.31**	45.13	34.01	09.62	42.33	44.40	05.97	**37.98**	32.82	17.57	22.14
Phi-4 14B	33.74	27.90	**37.47**	**37.52**	**47.46**	**60.62**	**14.29**	37.88	**37.00**	**22.71**	22.14
Mistral 7B	02.95	01.03	05.85	01.12	09.89	22.32	05.08	11.38	06.89	05.14	**25.69**
				LoRA Fine Tuning							
Qwen-2.5-7B	61.65	68.28	44.91	57.14	63.83	72.96	20.59	59.92	55.62	46.91	11.98
Gemma-3-12B	65.76	**71.54**	**47.13**	59.99	66.38	77.42	18.09	**63.17**	**58.04**	48.81	**13.68**
Llama-3.1-8B	**65.87**	70.79	45.35	58.84	64.93	**77.87**	11.59	62.41	56.46	46.70	11.38
Phi-4 14B	61.81	68.75	41.02	54.21	61.49	76.02	**22.73**	58.90	55.15	45.21	12.21
Mistral 7B	65.10	68.63	44.63	**61.10**	**66.50**	74.85	**22.73**	58.90	55.15	**50.06**	11.65

Table 11: Performance Analysis of the models on Target Groups. Org, Rel, and Pol refer to Organization, Religion, and Political, respectively. Color marks the highest performance for each configuration and metric.

Under this setting, all models exhibit improved balance between precision and recall. Qwen-2.5 7B and Gemma-3 12B also deliver strong results, both surpassing the 85% F1 threshold for the Non-Hate class and exceeding 81% in the Hate class.

5.2 Hate Category Detection

Table 10 shows the performance across different hate categories, the previous three settings.

5.2.1 Zero-Shot Prompting

Gemma-3 12B delivers the strongest overall performance, with the highest F1 scores across most hate categories, including Personal Offense (66.43%), Abusive/Violence (50.05%), and Religious (64.29%), as well as leading in both macro (42.66%) and micro (51.25%) averages. LLaMA-3.1-8B shows good performance in the Gender (46.94%) and Body Shaming (33.47%) cat-

Models	Non-Hate			Hate		
	P	R	F1	P	R	F1
Zero Shot Prompt						
Qwen-2.5-7B	64.52	93.39	76.31	85.64	43.45	57.61
Gemma-3-12B	**79.74**	90.47	**84.76**	87.66	74.66	**80.64**
Llama-3.1-8B	57.22	82.98	67.74	83.39	57.94	68.37
Phi-4 14B	35.42	88.44	50.58	71.76	**82.58**	18.27
Mistral 7B	65.05	**97.62**	78.08	**97.08**	60.15	74.28
Gemini 2.5	68.65	96.23	80.42	93.06	56.34	69.72
GPT 4o	73.15	95.27	83.10	93.22	65.14	77.26
Chain of Thought						
Qwen-2.5-7B	58.25	98.41	73.18	92.10	20.84	33.99
Gemma-3-12B	79.36	90.12	**84.40**	87.19	74.17	**80.15**
Llama-3.1-8B	75.45	87.29	80.94	83.06	68.69	75.19
Phi-4 14B	63.42	**96.36**	76.50	90.64	38.78	54.32
Mistral 7B	**93.91**	26.17	40.94	54.77	**98.14**	70.31
Gemini 2.5	70.88	96.04	82.68	93.10	57.82	70.53
GPT 4o	74.39	96.04	83.46	**93.17**	68.82	79.27
LoRA Fine Tuning						
Qwen-2.5-7B	81.07	90.29	85.43	87.84	76.89	82.00
Gemma-3-12B	81.49	89.34	85.21	86.87	77.53	81.89
Llama-3.1-8B	**84.31**	87.67	**85.96**	85.71	**81.94**	**83.83**
Phi-4 14B	79.42	**90.73**	84.70	**87.86**	74.04	80.36
Mistral 7B	81.14	88.67	84.74	86.08	77.26	81.43

Table 12: Benchmarking results of open and closed source models on the test split of BanHate. P, R, and F1 represent Precision, Recall, and F1 Score, respectively. Color marks the highest performance for each configuration and metric.

egories but exhibits lower overall accuracy. Phi-4 14B performs well in Political (49.78%) and Origin (28.24%) hate categories, though its performance remains inconsistent across other categories. Qwen-2.5-7B and Mistral 7B record comparatively weaker results, particularly in maintaining subset accuracy and categorical coherence.

5.2.2 Chain-of-Thought (CoT) Prompting

The performance in this configuration is more uneven than in previous setups. Gemma-3 12B again leads with consistently strong results across multiple hate categories. LLaMA-3.1 8B shows targeted gains in Origin (34.06%) and Religious (54.86%) hate detection, though its subset accuracy remains unstable. Mistral 7B excels in Personal Offence (67.26%), highlighting category-specific strength but limited generalization. Qwen-2.5 7B and Phi-4 14B show overall declines across most metrics under this configuration.

5.2.3 LoRA Fine-Tuning

Similar to the hate speech detection results, fine-tuning delivers the most consistent and substantial gains across all models. Gemma-3 12B and LLaMA-3.1 8B clearly dominate, leading in most categories and overall metrics. LLaMA-3.1 8B excels in Political (54.29%), Origin (40.51%), and Body Shaming (23.26%) detection, while Gemma-3 12B shows strong proficiency in Personal Offence (63.81%), Abusive/Violent (55.36%), Gender (56.78%), and Religious (71.33%) hate. All models achieve higher macro/micro F1 and accuracy, highlighting the clear advantage of task-specific fine-tuning. Mistral shows improved subset classification via a higher Hamming score but still lags behind in overall accuracy.

5.3 Target Group Detection

Table 11 reports the performances of the models in different target groups, and overall evaluation metrics in three settings: Zero-Shot, Chain-of-Thought (CoT), and LoRA Fine-Tuning.

5.3.1 Zero Shot Prompting

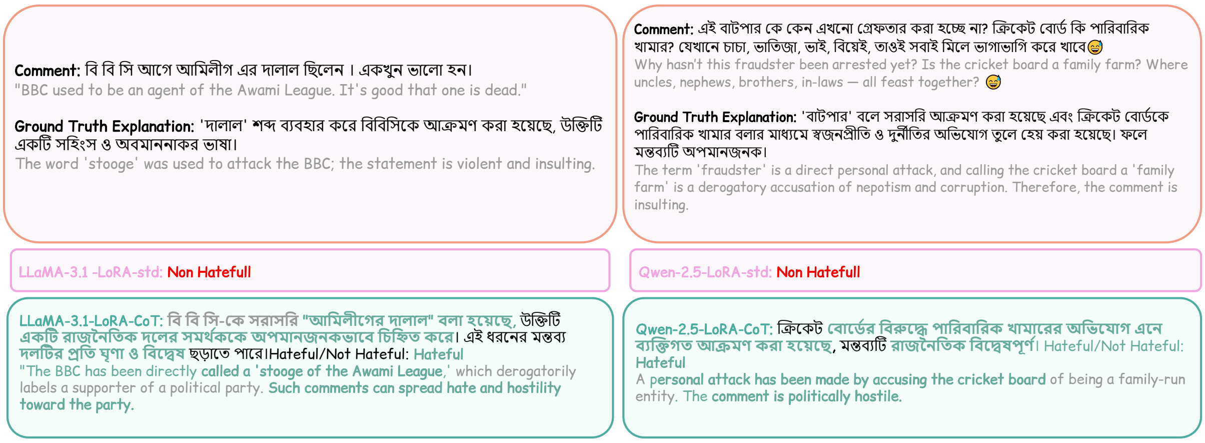
Gemma-3 12B delivers the strongest overall performance, leading most categories: Male (54.71%), Country (50.25%), Group (34.25%), and Religion (43.68%), while also achieving the highest macro (41.68%) and micro (38.66%) F1 scores. LLaMA-3.1 8B and Phi-4 14B follow, with Phi-4 showing exceptional strength in Organization (40.25%) and Female (67.90%) detection. Mistral 7B and Qwen-2.5 7B rank lower overall, though Qwen-2.5 attains top accuracy in select subsets.

5.3.2 Chain-of-Thought (CoT) Prompting

Phi-4 14B leads in multiple categories: Group (37.47%), Organization (37.52%), Country (47.46%), Religion (60.62%), and Political (14.29%), while attaining the highest macro-average (37.00%). LLaMA-3.1 8B shows consistent gains across categories and achieves the best score in the Male category (48.31%). Gemma-3 12B performs unevenly, excelling in the Female (57.04%) category but underperforming elsewhere. Mistral 7B consistently ranks last on all metrics, except for the hamming score.

5.3.3 LoRA Fine-Tuning

Similar to the previous analyses, all models show consistent improvement across categories. Gemma-3 12B and LLaMA-3.1 8B excel in the Female (71.54%), Group (47.13%), and Religion (77.87%) categories and Macro F1, Mistral 7B in the Organization (61.10%) and Country (66.50%) categories, and Subset Accuracy (50.06%). Phi-4 14B performs best in Political (22.73%) classification but trails the others in overall accuracy.



Comment: বি বি সি আগে আমিলীগ এর দালাল ছিলেন । একখুন ভালো হন।
"BBC used to be an agent of the Awami League. It's good that one is dead."

Ground Truth Explanation: 'দালাল' শব্দ ব্যবহার করে বিবিসিকে আক্রমণ করা হয়েছে, উক্তিটি একটি সহিংস ও অবমাননাকর ভাষা।
The word 'stooge' was used to attack the BBC; the statement is violent and insulting.

LLaMA-3.1 -LoRA-std: Non Hatefull

LLaMA-3.1-LoRA-CoT: বি বি সি-কে সরাসরি **"আমিলীগের দালাল" বলা হয়েছে,** উক্তিটি **একটি রাজনৈতিক দলের সমর্থককে অপমানজনকভাবে চিহ্নিত করে**। এই ধরনের মন্তব্য **দলটির প্রতি ঘৃণা ও বিদ্বেষ** ছড়াতে পারে।Hateful/Not Hateful: Hateful
"The BBC has been directly **called a 'stooge of the Awami League,'** which derogatorily labels a supporter of a political party. **Such comments can spread hate and hostility toward the party.**

Comment: এই বাটপার কে কেন এখনো গ্রেফতার করা হচ্ছে না? ক্রিকেট বোর্ড কি পারিবারিক খামার? যেখানে চাচা, ভাতিজা, ভাই, বিয়েই, তাওই সবাই মিলে ভাগাভাগি করে খাবে😅
Why hasn't this fraudster been arrested yet? Is the cricket board a family farm? Where uncles, nephews, brothers, in-laws — all feast together? 😅

Ground Truth Explanation: 'বাটপার' বলে সরাসরি আক্রমণ করা হয়েছে এবং ক্রিকেট বোর্ডকে পারিবারিক খামার বলার মাধ্যমে স্বজনপ্রীতি ও দুর্নীতির অভিযোগ তুলে হেয় করা হয়েছে। ফলে মন্তব্যটি অপমানজনক।
The term 'fraudster' is a direct personal attack, and calling the cricket board a 'family farm' is a derogatory accusation of nepotism and corruption. Therefore, the comment is insulting.

Qwen-2.5-LoRA-std: Non Hatefull

Qwen-2.5-LoRA-CoT: ক্রিকেট **বোর্ডের বিরুদ্ধে পারিবারিক খামারের অভিযোগ এনে ব্যক্তিগত আক্রমণ করা হয়েছে**, মন্তব্যটি **রাজনৈতিক বিদ্বেষপূর্ণ**। Hateful/Not Hateful: **Hateful**
A **personal attack has been made by accusing the cricket board** of being a family-run entity. The **comment is politically hostile.**

Figure 3: Error analysis of the LoRA finetuned models on Bengali hate speech detection. Detailed and accurate parts are emphasized in green and bold letters. The mistakes are highlighted in red.

5.4 Closed Source Models

Table 12 compares Gemini 2.5 Flash and GPT-4o under Zero-Shot and Chain-of-Thought (CoT) prompting for Hate vs. Non-Hate classification, using F1 scores as the primary metric. In the Zero-Shot Prompting, GPT-4o outperforms Gemini 2.5 with higher F1 scores in both Non-Hate (83.10%) and Hate (77.26%) categories. Gemini's strongest metric is its high Non-Hate recall (96.23%), indicating strong sensitivity, but with reduced precision. Gemini's subpar Hate recall weakens its overall classification performance. Both models improve performance in Chain-of-Thought (CoT) prompting, but GPT-4o maintains its lead.

6 Qualitative Error Analysis

We find several notable patterns and recurring errors in hate speech detection, *e.g.*, Gemma-3 frequently misclassifies *aggressive or threatening* discussions as *peaceful and constructive*. This could be due to insufficient contextual cues for violence or aggression in the training data. We also hypothesize that this effect of inadequacy is somewhat mitigated when the models are fine-tuned. Figure 3 errors that persist in fine-tuned models, allowing us to analyze beyond the training distributional biases.

In Figure 3, we observe LLaMA-3.1 and Qwen-2.5 models in zero-shot prompting misclassifies texts as non-hateful. However, the chain of thought prompting enables the models to correctly classify the texts as hateful while producing proper explanations against the claims. Hence, LoRA fine-tuning combined with chain of thought prompting minimizes the errors produced by the models.

7 Discussion: Pretraining Distribution

Variations in model performance are closely tied to the underlying distribution of the pretraining data. Models exposed to a larger amount of Bangla text should develop a better linguistic understanding, while insufficient data prevents the model from capturing the nuances of hate speech. The *nature* of the data can also play a critical role, *e.g.*, real-world and updated Bangla sources may generalize better than an outdated corpus in limited settings. Unfortunately, analyzing the attribution of performance variation to the training data is infeasible as none of the evaluated models publicly disclose their pretraining data. The opacity highlights a broader limitation of interpreting model behaviour in low-resource languages.

8 Conclusion

We present BANHATE, a Bangla hate-speech dataset spanning 19,203 YouTube comments from April 2024 to June 2025 across five content domains and seven hate categories/target groups. The dataset enables classification training and evaluation. Using a diverse suite of open and closed-source LLMs under zero-shot, chain-of-thought prompting, and LoRA fine-tuning strategies, we showed that prompting and fine-tuning strongly influence detection performance. LoRA fine-tuning delivers consistent gains in both hate and non-hate F1. We hope that our dataset and findings will be a valuable resource for future research on low-resource languages such as Bangla.

Limitations

BanHate comprises only YouTube top-level comments collected between April 2024 and June 2025, limiting the validity of the dataset to a single platform only. Our evaluation reports results from a single LoRA epoch with greedy decoding, lacking a held-out development set or multi-seed runs.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Md. Tofael Ahmed, Maqsudur Rahman, Shafayet Nur, Azm Islam, and Dipankar Das. 2021. Deployment of machine learning and deep learning algorithms in detecting cyberbullying in bangla and romanized bangla text: A comparative study. In *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pages 1–10.

Sai Suman Aluru, Binny Mathew, Punyajoy Saha, and Animesh Mukherjee. 2020. Deep learning models for multilingual hate speech detection. *arXiv preprint arXiv:2004.06465*.

Tasnim Tarannum Aurpa, Rifat Sadik, and Md Saiful Ahmed. 2022. Abusive bangla comments detection on facebook using transformer-based deep learning models. *Social Network Analysis and Mining*, 12(1):24.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th international conference on World Wide Web companion*, pages 759–760.

Deeparghya Dutta Barua, MSUR Sourove, Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Md Fahim, and Farhad Alam Bhuiyan. 2024. Penta ml at exist 2024: tagging sexism in online multimodal content with attention-enhanced modal context. *Working Notes of CLEF*.

Tanveer Ahmed Belal, G. M. Shahariar, and Md. Hasanul Kabir. 2023. Interpretable multi labeled bengali toxic comments classification using deep learning. In *2023 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–6.

Fariha Tahosin Boishakhi, Ponkoj Chandra Shill, and Md. Golam Rabiul Alam. 2021. Multi-modal hate speech detection using machine learning. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 4496–4499.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in Bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Estiak Ahmed Emon, Shihab Rahman, Joti Banarjee, Amit Kumar Das, and Tanni Mittra. 2019. A deep learning approach to detect abusive bengali text. In *2019 7th International Conference on Smart Computing and Communications (ICSCC)*, pages 1–5.

Vaibhav Goel, Dhruv Sahnan, Saptarshi Dutta, Anil Bandhakavi, and Tanmoy Chakraborty. 2023. Hatemongers ride on echo chambers to escalate hate speech diffusion. arXiv preprint arXiv:2302.02479. https://arxiv.org/abs/2302.02479.

Shilpa Gudka, Felix Reer, and Thorsten Quandt. 2023. Towards a framework for flourishing through social media: A systematic review of 118 research studies. *Current Opinion in Psychology*, 53:101633.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Md Sakib Ul Rahman Sourove, Deeparghya Dutta Barua, Md Fahim, and Md Farhad Alam Bhuiyan. 2025. BanTH: A multi-label hate speech detection dataset for transliterated Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7217–7236, Albuquerque, New Mexico. Association for Computational Linguistics.

E. Hashmi and S. Y. Yayilgan. 2024. Multi-class Hate Speech Detection in the Norwegian Language Using FAST-RNN and Multilingual Fine-Tuned Transformers. *Complex Intelligent Systems*, 10:4535–4556.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Refa Annisatul Ilma, Setiawan Hadi, and Afrida Helen. 2021. Twitter's hate speech multi-label classification using bidirectional long short-term memory (bilstm) method. In *2021 International Conference on Artificial Intelligence and Big Data Analytics*, pages 93–99.

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.

Sarif Sultan Saruar Jahan, Raqeebir Rab, Peom Dutta, Hossain Muhammad Mahdi Hassan Khan, Muhammad Shahariar Karim Badhon, Sumaiya Binte Hassan, and Ashikur Rahman. 2023. Deep learning based misogynistic bangla text identification from social media. *Computing and Informatics*, 42(4):993–1012.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Andreas M. Kaplan and Michael Haenlein. 2010. Users of the world, unite! the challenges and opportunities of social media. *Business Horizons*, 53(1):59–68.

Youngwook Kim, Shinwoo Park, and Yo-Sub Han. 2022. Generalizable implicit hate speech detection using contrastive learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6667–6679, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. 2019. Hate speech detection: Challenges and solutions. *PloS one*, 14(8):e0221152.

Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song, and Dit-Yan Yeung. 2019. Multilingual and multi-aspect hate speech analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4675–4684, Hong Kong, China. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

R Reghunathan and AS Asha. 2022. Hate speech detection in conventional language on social media by using machine learning. In *International Journal of Engineering Research & Technology (IJERT)*, volume 11.

Nasif Istiak Remon, Nafisa Hasan Tuli, and Ranit Debnath Akash. 2022. Bengali hate speech detection in public facebook pages. In *2022 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 169–173.

Nahian Romim, Muhtasim Ahmed, Hasib Talukder, and Md Shamsul Islam. 2020. Hate speech detection in the bengali language: A dataset and its baseline evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*.

Md. Saifuddin, Mohiuddin Ahmed, Spandan Basu, and Pritam Acharjee. 2023. Enhancing online safety: Natural language processing based multi-label cyberbullying classification in bangla. In *2023 26th International Conference on Computer and Information Technology (ICCIT)*, pages 1–6.

Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2022. M-bad: A multilabel dataset for detecting aggressive texts and their targets. In *Proceedings of the Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situations*, pages 75–85.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

The Havok Journal. 2023. The psychology behind social media hate. `https://havokjournal.com/culture/the-psychology-behind-social-media-hate/`.

Sinéad Walsh and Paul Greaney. 2025. Multiclass hate speech detection with an aggregated dataset. *Natural Language Processing*, page 1–17.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wikipedia, Ethnologue. 2025. Bengali language speaker statistics. `https://en.wikipedia.org/wiki/Bengali_language`. Accessed: July 27, 2025.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of

100+ language models. In *Proceedings of the 62nd ACL*, Bangkok, Thailand. Association for Computational Linguistics.

A Annotation Guidelines

Annotators were instructed to evaluate each comment to determine whether it constitutes hate speech. If so, they identified the specific target group, assigned one or more relevant hate categories. Each comment was evaluated within its broader context, including metadata such as video title, publication date, and thematic category, *e.g.*, News & Politics, Entertainment. These annotation instructions were designed to ensure high inter-annotator agreement and encourage objective and consistent judgments.

A.1 General Annotation Instructions

- **Target-Orientation:** Each comment was annotated by identifying the target group(s) from a predefined list: *Male, Female, Group, Organization, Country, Religious, Politics*.

- **Hate-Label Classification:** Each comment can be assigned to one or multiple hate categories, *e.g.*, Religious, Gender, Body Shaming, Abusive/Violence.

- **Context-Aware Interpretation:** Annotators were instructed to interpret intent and implicit meaning, *e.g.*, when sarcasm, metaphors, or culturally coded language were used.

- **Bias-Free and Objective Labeling:** Annotators maintained neutrality and judged based on harm, discrimination, or dehumanization rather than personal opinion.

- **Annotation Redundancy and Verification:** Each comment was annotated by three individuals, with expert adjudication and discussions resolving conflicts.

- **Mental Well-being:** Annotators took 5–10 minute breaks each hour to reduce cognitive fatigue.

- **Cultural Sensitivity and Localization:** Annotators were guided to identify culturally specific hate forms, *e.g.*, regional slurs, gendered insults, or religious undertones in Bangla conversations.

A.2 Binary Classification – Hate/Non-Hate

- **Hate Speech (H):** A comment should be labeled as Hate Speech if it expresses prejudiced, offensive, abusive, or harmful language directed at identity, groups, or individuals based on gender, religion, nationality, ethnicity, political affiliation, and similar factors.

- **Non-Hate Speech (NH):** Comments that lack hostile, discriminatory, or dehumanizing content. This category may include critical or sarcastic expressions that do not constitute targeted attacks.

A.3 Hate Speech Classification

Annotators labeled each hate speech comment with one of the following categories.

Political: Abusive language or incitement of harm directed at political ideologies, parties, or figures.

Religious: Discriminatory or divisive attacks against religious groups, beliefs, or practices.

Gender: Hostility based on gender identity or stereotypes, including, but not limited to, sexism and transphobia.

Personal Offense: Degrading attacks or insults targeting an individual rather than a group.

Abusive/Violence: Explicit threats or incitement to physical violence or harm.

Body Shaming: Criticism or mockery of physical appearance, body shape, or disabilities.

B Prompt

All the prompts are detailed in this section.

Zero Shot Prompt Used for Prediction

You are given a Bangla text. Determine if it is "Hate" or "Non Hate".
If it is hate speech, identify the category of hate and the target group, choosing **only from the following options**:

Hate_Category:

```
['Abusive/Violence', 'Body Shaming', 'Gender', 'Origin',
'Personal Offence', 'Political', 'Religious']
```

Target_Group:

```
['Country', 'Female', 'Group', 'Male', 'Organization',
'Politics', 'Religious']
```

Return your answer in this strict JSON format:

```
{
  "first_answer":"Hate"or"Non Hate",
  "final_answer":"Hate"or"Non Hate",
  "hate_category": [ ... ],
  "target_group": [ ... ]
}
```

Chain of Thought Prompt Used for Prediction

You are tasked with detecting hate speech in Bangla text using a step-by-step reasoning approach.

Follow these steps carefully:

1. Understand the **literal and implicit meaning** of the Bangla text.
2. Analyze the **tone and intent**: Is the speaker expressing hostility, discrimination, or inciting hatred?
3. Look for **contextual cues**: Are there slurs, derogatory phrases, targeted groups, or implied harm?
4. Make a **final decision** based on your analysis.
5. **Explain** your reasoning briefly in Bangla.

Now, respond using the following strict JSON format:

```
 {
"thought_process": "Your step-by-step reasoning in English",
"is_hate": "Hate" or "Non Hate",
}
```

BOIGENRE: A Large-Scale Bangla Dataset for Genre Classification from Book Summaries

Rafi Hassan Chowdhury, Rahanuma Ryaan Ferdous
Islamic University of Technology
{rafihassan, rahanumaryaan}@iut-dhaka.edu

Abstract

The classification of literary genres plays a vital role in digital humanities and natural language processing (NLP), supporting tasks such as content organization, recommendation, and linguistic analysis. However, progress for the Bangla language remains limited due to the lack of large, structured datasets. To address this gap, we present BOIGENRE, the first large-scale dataset for Bangla book genre classification, built from publicly available summaries. The dataset contains 25,951 unique samples across 16 genres, showcasing diversity in narrative style, vocabulary, and linguistic expression. We provide statistical insights into text length, lexical richness, and cross-genre vocabulary overlap. To establish benchmarks, we evaluate traditional machine learning, neural, and transformer-based models. Results show that while unigram-based classifiers perform reasonably, transformer models, particularly BanglaBERT, achieve the highest F1-score of 69.62%. By releasing BOIGENRE and baseline results, we offer a valuable resource and foundation for future research in Bangla text classification and low-resource NLP. Our dataset and code is publicly available at https://github.com/Grumpy-Frog/BoiGenre

1 Introduction

Automatic classification of books by genre enhances digital library management and recommendation systems by aligning content with readers' interests. It enables personalized suggestions, improves user experience, and helps publishers understand audience preferences. However, most studies focus on English, leaving a significant gap for Bangla despite its global prominence. This gap stems from the lack of high-quality Bangla datasets, limiting the use of modern NLP techniques. Developing resources like BOIGENRE can bridge this gap and advance Bangla language research.

With the rise of modern machine learning frameworks, automated genre classification has become increasingly effective. This task involves identifying the genre of a book, movie, or artwork based on features such as the title, summary, or cover image. Previous studies have combined textual features like titles and summaries for genre prediction (Adhikari, 2025), while others explored language-specific datasets such as PPORTAL—the Portuguese literary dataset (Scofield et al., 2022). Moreover, genre detection has also been approached through visual data, as seen in the IMDb book cover dataset (Buczkowski et al., 2018).

Despite these developments, resources for Bangla literature remain limited. Existing datasets are often small or restricted to a few genres, reducing model generalization (Sethy et al., 2023). Furthermore, many studies have not leveraged transformer-based architectures such as BERT, which excel at contextual understanding. To address these gaps, we introduce the BOIGENRE dataset—a large-scale and diverse Bangla book genre dataset—and evaluate it using traditional machine learning, deep learning, and transformer models to establish strong baselines for future research.

In this paper, we introduce BOIGENRE, the first large-scale dataset for Bangla book genre classification based on summaries. This dataset addresses the scarcity of Bangla resources in natural language processing and reflects the diversity of Bangla literature. We evaluate multiple models, from traditional machine learning to transformer-based architectures, establishing strong baselines for future research. Future work may focus on expanding the dataset, incorporating full texts, and exploring advanced transformer or cross-lingual models to further enhance Bangla genre classification.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 249–258
December 23, 2025 ©2025 Association for Computational Linguistics

2 Literature Review

Most research on book genre classification has been conducted on English datasets such as Goodreads and Project Gutenberg. These studies primarily rely on user tags, full text, or book covers for genre detection (McAuley et al., 2017; Iwana et al., 2016). Some works have also used summaries, such as Bhuiyan et al. (Bhuiyan et al., 2023) and Adhikari (Adhikari, 2025), but these are based on small English datasets with limited genre coverage. In another direction, Pasha et al. (Pasha et al., 2023) focused on Bangla poem classification, though their work was restricted to only two genres and did not use book summaries. In addition, multilingual and cross-lingual approaches have been explored, including XLM-RoBERTa-based models for multiple languages (Classla, 2024), cross-lingual summarization (Zhang et al., 2024), and universal cross-lingual text classification (Savant, 2024), showing that genre detection can benefit from transfer learning across languages.

Over the years, methods for genre detection have evolved from simple TF-IDF and SVMs to deep learning models and transformer-based approaches like BERT. While these models show better performance, most prior works remain limited to English datasets and face issues such as label noise, class imbalance, and unclear genre definitions (Iwana et al., 2016; Kabir et al., 2023). In Bangla, resources are especially scarce, with existing datasets focusing mainly on sentiment analysis rather than genre classification (Kabir et al., 2023; Islam et al., 2021; Biswas, 2025; Alvi et al., 2022; Mahmud and Mahmud, 2024; Ahmed et al., 2023).

Our dataset, BOIGENRE, addresses these gaps by providing Bangla book summaries with well-defined genre labels. Unlike noisy user tags or cover images, summaries capture the narrative context more directly. This makes the dataset suitable for supervised genre detection and contributes a new resource for Bangla, which has been underrepresented in this research area (Kabir et al., 2023).

3 The BOIGENRE Dataset

In this section, we describe the process of curating the proposed BOIGENRE dataset. We outline the data sources, collection strategy, and refinement steps. These details provide the foundation for its use in downstream experiments.

3.1 Data Collection

We constructed our dataset by collecting information on books from the online bookstore Rokomari.[1] In total, we gathered metadata for 25,951 books spanning 16 distinct genres. The data collection process was carried out using the `BeautifulSoup` library. We first compile a list of URLs corresponding to the 16 genre categories available on the platform and then iteratively scraped the details of each book from these pages. Specifically, we scraped the *title*, *author*, *genre*, and *summary* fields, which together provide a comprehensive representation of the metadata of each book. This information forms the backbone of the dataset, as it allows for both high-level categorization across genres and fine-grained analysis based on author contributions and narrative content. By covering a wide range of genres and including summaries alongside bibliographic details, the dataset captures a diverse cross section of Bangla literature and creates opportunities for multiple downstream natural language processing tasks.

3.2 Preprocessing

Processing Step	Samples Remaining
Initial collection	98,811
After removing missing summaries	46,677
After removing duplicates	25,951

Table 1: Number of samples retained after each data pre-processing step.

The initial collection resulted in a total of 98,811 samples. However, a large portion of these records were incomplete, particularly missing summaries. After removing all samples without summaries, we retained 46,677 valid entries. Next, we identified and eliminated duplicate instances, where both the title of the book and the summary were identical across multiple records. This cleaning step was crucial to reduce redundancy and ensure dataset quality. Following the de-duplication process, the final dataset consisted of 25,951 unique book samples, which form the basis for our subsequent experiments and analyzes 1.

4 Statistical Analysis

Table 2 and related figures provide an overview of the statistical properties of the BOIGENRE dataset across genres. The distribution of samples reveals

[1] https://www.rokomari.com

Genre	**N**	**Words / Summ.**	**Sents / Summ.**	**TTR**	**Vocab Size**
Biography and Autobiography	5560	160.35±148.27	11.84±11.60	0.093	83148
Contemporary Novel	4669	136.17±89.32	14.21±10.61	0.103	65595
History and Tradition	3224	165.86±126.55	11.39±9.66	0.108	57626
Religious	2674	170.14±193.39	11.62±10.98	0.102	46290
Contemporary Story	2234	127.43±96.47	11.77±10.19	0.142	40506
Classic Novel	1189	139.59±112.08	13.56±11.91	0.183	30294
Thriller	1066	135.07±82.66	13.64±9.38	0.181	26125
Science Fiction	854	132.39±94.35	13.10±9.60	0.189	21348
Shishu Kishor	803	126.68±103.89	14.10±11.80	0.192	19567
Politics	778	172.64±159.49	11.36±10.34	0.175	23476
Philosophy	752	155.39±125.05	10.47±8.37	0.188	21991
Mystery	719	131.71±81.12	12.50±8.41	0.216	20497
Classic Story	593	135.25±189.75	12.08±21.28	0.246	19747
Adventure	354	140.39±132.46	14.01±12.03	0.261	12950
Math	267	178.46±171.03	11.67±10.93	0.178	8499
Cooking, Food and Nutrition	215	194.84±190.98	12.14±9.03	0.221	9274

Table 2: Summary statistics across genres. N is the number of summaries, *Words / Summ.* is the average number of words per summary (mean±std), *Sents / Summ.* is the average number of sentences per summary (mean±std), *TTR* is the type–token ratio (lexical diversity), and *Vocab Size* is the number of unique word types observed in each genre.

that *Biography and Autobiography* is the most well-represented category with 5,560 summaries, followed by *Contemporary Novel* with 4,669 and *History and Tradition* with 3,224. In contrast, specialized domains such as *Math* and *Cooking, Food and Nutrition* contain far fewer samples, only 267 and 215 respectively, indicating a strong imbalance across genres that realistically reflects the Bangla publishing landscape. Such imbalance also introduces challenges for downstream modeling, as classifiers must learn to generalize across both abundant and scarce categories without biasing predictions toward majority classes.

When looking at text length, notable differences emerge. Summaries in *Cooking, Food and Nutrition* and *Math* are the longest on average, exceeding 175 words per summary, while *Shishu Kishor* and *Contemporary Story* contain much shorter summaries, around 127 words. Interestingly, sentence-level statistics suggest that *Contemporary Novel* and *Shishu Kishor* have more segmented narrative structures, with around 14 sentences per summary, compared to only 10–12 sentences in most other genres. This aligns with the expectation that children's and contemporary fiction are written in shorter, more digestible units.

Lexical diversity, measured using type–token ratio (TTR), varies across genres. Genres with fewer samples, such as *Adventure* with a TTR of 0.261 and *Classic Story* with a TTR of 0.246, display the highest lexical diversity, indicating more varied vocabulary relative to their size. Larger domains like *Biography and Autobiography* with a TTR of 0.093 and *Religious* with a TTR of 0.102 show lower diversity, likely due to repeated thematic expressions. Broad domains such as *Biography and Autobiography* and *Contemporary Novel* contribute the largest vocabularies with over 65,000–83,000 unique types, while narrow domains like *Math* and *Cooking, Food and Nutrition* have smaller vocabularies, under 10,000 types.

These findings suggest that the BOIGENRE dataset captures a wide spectrum of Bangla literary styles, ranging from highly narrative forms to compact, domain-specific texts. At the same time, the inherent class imbalance poses a challenge, motivating the development of methods that can better handle skewed distributions and improve performance on minority genres.

5 Developing Benchmark for BOIGENRE

To provide a benchmark for Bangla literary text classification, we conduct experiments on the BOIGENRE dataset. The collection covers a wide spectrum of genres and exhibits substantial diversity in narrative style, linguistic expression, and vocabulary usage. Such variety makes BOIGENRE a suitable resource for testing the capacity of different NLP models to generalize across heterogeneous text domains.

Model	Feature Set	Vectorizer	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
SVM	Unigram	TF–IDF	63.58	62.88	63.58	63.02
Logistic Regression	Unigram	TF–IDF	61.00	63.10	61.00	61.40
Naive Bayes	Unigram	BoW	60.67	65.15	60.67	57.50
Gradient Boosting	Unigram	TF–IDF	55.47	55.04	55.47	53.54
SVM	Bigram	TF–IDF	54.10	52.88	54.10	53.21
SVM	2+3gram	TF–IDF	53.28	52.15	53.28	52.49
Naive Bayes	Bigram	BoW	55.84	57.15	55.84	52.05
Naive Bayes	2+3gram	BoW	54.76	55.54	54.76	51.11
Logistic Regression	2+3gram	TF–IDF	51.08	52.28	51.08	51.07
Random Forest	Unigram	TF–IDF	54.35	65.83	54.35	48.71
BiLSTM	Unigram	fastText	57.17	57.62	57.17	55.84
Custom CNN	Unigram	fastText	60.79	59.26	60.79	59.23
BanglaBERT	Contextual subword embeddings	BanglaBERT tokenizer	**69.34**	**70.26**	**69.34**	**69.62**

Table 3: Performance comparison of traditional machine learning models, deep learning architectures, and pretrained transformers for Bangla text classification. Accuracy, Precision, Recall, and F1 Score are reported as weighted averages across classes. Best results are highlighted in bold.

5.1 Baseline Models and Features

To establish benchmark results on the BOIGENRE dataset, we experiment with a diverse set of models, ranging from classical machine learning classifiers to neural architectures and pretrained transformers. This setup enables us to analyze the relative effectiveness of shallow lexical features versus contextualized embeddings in Bangla text classification.

For lexical features, we extract unigram and higher-order n-gram (bi- and tri-gram) representations at both the word and character levels. These features are vectorized using TF–IDF and Bag-of-Words (BoW), two widely used representations in text classification tasks. Classical classifiers, including Support Vector Machines (SVM) (Cortes and Vapnik, 1995), Logistic Regression (LR) (Le Cessie and Van Houwelingen, 1992), Naive Bayes (NB) (John and Langley, 1995), Random Forest (RF) (Breiman, 2001), and Gradient Boosting (GB) (Friedman, 2001), are trained on these feature sets. These models serve as lightweight, interpretable baselines that capture surface-level lexical correlations.

To incorporate distributed representations, we utilize fastText embeddings (Joulin et al., 2017) as inputs to neural models. A BiLSTM (Hochreiter and Schmidhuber, 1997) is employed to capture long-range sequential dependencies, while a custom CNN (Kim, 2014) is used to detect local compositional patterns. Such architectures have shown effectiveness in morphologically rich languages, where subword-level modeling is essential.

We further evaluate a transformer-based model, BanglaBERT (Bhattacharjee et al., 2022), pretrained on large-scale Bangla corpora. As a contextualized encoder, BanglaBERT captures both semantic and syntactic information and represents the strongest baseline in our study.

For evaluation, we report Accuracy, Precision, Recall, and F1-score. Given the class imbalance in BOIGENRE, the weighted F1-score is emphasized as the primary metric, as it balances precision and recall across unevenly distributed categories (Sokolova et al., 2006).

5.2 Results & Findings

Table 3 reports the performance of traditional machine learning models, neural architectures, and pretrained transformers on the BOIGENRE dataset. *BanglaBERT* achieves the best results with 69.34% accuracy and 69.62% F1, outperforming both classical and deep learning baselines. Since the dataset is class-imbalanced, the F1 score is a more reliable indicator of performance, and BanglaBERT consistently leads across all metrics.

Among classical approaches, SVM with unigram and TF–IDF performs strongest with 63.58% accuracy, while higher-order n-grams degrade performance due to sparsity. Naive Bayes performs competitively on unigrams but shows weaker F1 scores, reflecting difficulty in handling minority genres. Tree-based ensembles such as Random Forest and Gradient Boosting perform worst, struggling with sparse, high-dimensional features.

Neural models using fastText embeddings, such as BiLSTM and a custom CNN, provide modest

Genre	Precision (%)	Recall (%)	F1 Score (%)
Adventure	40.85	50.00	44.96
Biography and Autobiography	79.52	77.05	78.26
Classic Novel	41.60	50.00	45.41
Classic Story	20.49	32.47	25.13
Contemporary Novel	75.19	67.05	70.89
Contemporary Story	60.53	66.59	63.41
Cooking, Food and Nutrition	100.00	92.86	96.30
History and Tradition	73.87	77.50	75.64
Math	96.49	96.49	96.49
Mystery	47.89	59.65	53.12
Philosophy	67.41	62.33	64.77
Politics	42.86	42.33	42.59
Religious	80.30	83.56	81.90
Science Fiction	73.17	83.33	77.92
Shishu Kishor	61.49	50.28	55.32
Thriller	63.01	52.87	57.50

Table 4: Classwise analysis for BanglaBERT model.

gains over traditional baselines, but remain behind BanglaBERT. Their limitations suggest that sequential or convolutional architectures alone cannot capture the full morphological and semantic complexity of Bangla text.

The superiority of BanglaBERT can be attributed to its pretraining on large-scale Bangla corpora, which enables it to model polysemy, discourse-level structure, and genre-specific variation. At the same time, the relatively strong unigram-based linear classifiers highlight that surface lexical features remain effective for Bangla classification tasks. Together, these findings emphasize both the promise of pretrained transformers and the continued value of lightweight baselines for future work on Bangla NLP.

5.3 Error Analysis

As shown in Table 4, BanglaBERT achieves strong performance on high-resource genres such as Biography and Autobiography (F1 78.26) and Religious (F1 81.90), as well as on domain-specific but lexically narrow categories like Math and Cooking, Food and Nutrition, where F1-scores exceed 95. In contrast, minority genres with high lexical diversity such as Adventure with F1 44.96, Classic Story with F1 25.13, and Mystery with F1 53.12 remain more difficult. This highlights how class imbalance and high type–token ratio jointly limit generalization. Table 2 shows that Adventure has a TTR of 0.261 and Classic Story has a TTR of 0.246, indicating strong lexical variation despite very small sample sizes, which makes them harder to model consistently.

The word-level statistics in Appendix Table 6 reveal that frequent unigrams such as করে, এই, তার, and এক appear across many genres, offering limited discriminative value. Their dominance leads to frequent misclassification between closely related categories, most notably Contemporary Novel, Classic Novel, Classic Story, and Shishu Kishor. Appendix Figure 1 confirms this pattern, showing cosine similarity values above 0.90 among these genres, which indicates a high degree of lexical overlap. Such overlap makes surface word distributions insufficiently distinctive and explains why classifiers often confuse them. In contrast, domain-specific genres like Math and Cooking, Food and Nutrition exhibit much lower similarity with other categories, with cosine scores below 0.55. Despite having fewer samples, these genres achieve higher classification accuracy because their specialized vocabularies are more distinctive. These findings show that classification errors in BOIGENRE are shaped not only by class imbalance but also by substantial cross-genre vocabulary overlap, underscoring the need for models that can capture deeper semantic and contextual information beyond surface lexical similarity.

Genre	Sample Count
Biography and Autobiography	294
Politics	123
History and Tradition	41
Contemporary Story	3
Philosophy	2

Table 5: Genre-wise distribution of books associated with “বঙ্গবন্ধু” in the BOIGENRE dataset.

Another source of difficulty comes from thematic overlap across genres. Table 5 shows that books related to বঙ্গবন্ধু are distributed across Biography and Autobiography, Politics, and History and Tradition. Because thematically similar material is spread across multiple genres, the boundaries between classes are blurred, making it harder for the model to assign the correct label.

Taken together, these observations show that misclassification in BOIGENRE is shaped not only by class imbalance but also by high lexical diversity in low-resource genres and significant vocabulary overlap across thematically related categories. Future work can explore class-aware training strategies, targeted data augmentation, and representation learning approaches that capture deeper semantic and discourse-level patterns beyond surface lexical similarity.

6 Conclusion

In this paper, we introduce the first large-scale dataset BOIGENRE for classifying the genres of Bangla books based on summaries. This dataset represents an important step toward enriching Bangla resources in natural language processing and capturing the diversity of Bangla literature. We further evaluate a range of traditional, deep learning, and transformer-based models to establish strong baselines for future research. Future work may focus on expanding the dataset with additional sources and genres, incorporating full texts for deeper contextual understanding, and addressing class imbalance and linguistic variations. Moreover, exploring human evaluation methods and advanced transformer or cross-lingual models could further enhance the robustness and generalization of Bangla genre classification systems.

7 Limitations

Despite the contributions of this study, several limitations remain. The BOIGENRE dataset was compiled solely from the Rokomari website, which may not capture the full diversity of Bangla literature in terms of writing style, era or publication source. The dataset also exhibits class imbalance, as certain genres contain fewer samples than others, potentially affecting the model's ability to generalize across all categories. Furthermore, since the dataset is based only on book summaries rather than complete texts, it may lack deeper contextual or stylistic cues that are often essential for accurate genre identification. The current work focuses on sixteen primary genres, leaving out finer subgenres or overlapping categories that could provide more nuanced classification. Linguistic variations, such as regional dialects, informal spellings, and transliterated words, can also introduce inconsistencies within the data. In addition, the evaluation relied solely on quantitative performance metrics like Accuracy, Precision, Recall, and F1-score, without incorporating human judgment or interpretability analysis. Lastly, while transformer-based models such as BanglaBERT achieved strong performance, further experiments with multilingual or domain-specific transformers could provide deeper insights into model robustness and cross-domain generalization.

References

Jhimli Adhikari. 2025. Leveraging book genre classification using machine learning. *DESIDOC Journal of Library & Information Technology*, 45(3).

Zishan Ahmed, Shakib Sadat Shanto, and Akinul Islam Jony. 2023. Advancement in bangla sentiment analysis: A comparative study of transformer-based and transfer learning models for e-commerce sentiment classification. *Journal of Information Science and Engineering*, 15(6):48–63.

Nasif Alvi, Kamrul Hasan Talukder, and Abdul Hasib Uddin. 2022. Sentiment analysis of bangla text using gated recurrent neural network. In *International Conference on Innovative Computing and Communications*, pages 77–86. Springer.

Abhik Bhattacharjee, Shammur Absar Sarker, and et al. 2022. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7767–7778.

S.R. Bhuiyan, M.R.H. Khan, U.S. Afroz, and M.S. Rahman. 2023. Identifying genre of a book from its summary using machine learning approach. In *Proceedings of the International Conference on Advances in Computing, Communication and Control*. Springer.

Jahanur Biswas. 2025. Bangla sentiment dataset. Mendeley Data.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Przemyslaw Buczkowski, Antoni Sobkowicz, and Marek Kozlowski. 2018. Deep learning approaches towards book covers classification. In *ICPRAM*, pages 309–316.

Classla. 2024. Xlm-roberta base multilingual text genre classifier. Accessed: 2025-10-01.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Khondoker Islam and 1 others. 2021. Sentnob: A dataset for analysing sentiment on noisy bangla texts. Kaggle.

Brian Kenji Iwana, Syed Tahseen Raza Rizvi, Sheraz Ahmed, Andreas Dengel, and Seiichi Uchida. 2016. Judging a book by its cover. *arXiv preprint arXiv:1610.09204*.

George H John and Pat Langley. 1995. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431.

M. Kabir and 1 others. 2023. Banglabook: A large-scale bangla dataset for sentiment analysis from book reviews. `https://arxiv.org/abs/2305.12053`.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Saskia Le Cessie and Hans C Van Houwelingen. 1992. Ridge estimators in logistic regression. *Applied Statistics*, pages 191–201.

Hemal Mahmud and Hasan Mahmud. 2024. Enhancing sentiment analysis in bengali texts: A hybrid approach using lexicon-based algorithm and pretrained language model bangla-bert. *arXiv preprint arXiv:2411.19584*.

Julian McAuley and 1 others. 2017. Goodreads book graph datasets. `https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html`.

Syed Tangim Pasha, Ashraful Islam, Mohammed Masudur Rahman, Eshtiak Ahmed, and Zahangir Alam. 2023. Genre classification of bangla poem using machine learning and deep learning techniques. Technical report, Independent University, Bangladesh.

R. Savant. 2024. Universal cross-lingual text classification. *arXiv preprint arXiv:2406.11028*. Accessed: 2025-10-01.

Clarisse Scofield, Mariana O Silva, Luiza de Melo-Gomes, and Mirella M Moro. 2022. Book genre classification based on reviews of portuguese-language literature. In *International Conference on Computational Processing of the Portuguese Language*, pages 188–197. Springer.

Abhisek Sethy, Ajit Kumar Rout, Archana Uriti, and Surya Prakash Yalla. 2023. A comprehensive machine learning framework for automated book genre classifier. *Revue d'Intelligence Artificielle*, 37(3):745.

Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. 2006. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. *AI 2006: Advances in Artificial Intelligence*, pages 1015–1021.

R. Zhang and 1 others. 2024. Cross-lingual cross-temporal summarization: Dataset creation, modeling, and evaluation. *Computational Linguistics*, 50(3):1001–1023.

A Appendix

Genre	**10 Most Frequent Words**
Adventure	এক (491), করে (404), থেকে (361), আর (334), তার (311), এই (260), কিন্তু (200), একটা (197), এবং (192), করতে (172)
Biography and Autobiography	এই (7773), এবং (6355), তিনি (6264), করে (6255), তাঁর (5892), থেকে (5171), তার (4503), এক (3964), এর (3804), আর (3329)
Classic Novel	করে (1572), তার (1371), এই (1350), এক (1082), আর (1056), থেকে (895), আমার (774), এবং (716), কিন্তু (693), আমি (619)
Classic Story	গল্প (873), এই (693), করে (638), তার (540), এক (484), আর (453), থেকে (372), এবং (339), গল্পের (324), আছে (283)
Contemporary Novel	করে (6746), তার (5985), এই (5265), এক (4520), আর (4509), থেকে (3735), হয় (2771), হয়ে (2681), আমার (2498), কিন্তু (2489)
Contemporary Story	গল্প (3123), করে (2621), এই (2367), তার (1736), আর (1638), এক (1547), থেকে (1479), গল্পের (1456), আমার (1410), জীবনের (1259)
Cooking, Food and Nutrition	রান্না (438), এই (357), রান্নারের (329), করে (303), খাবার (229), চিকেন (225), করা (222), থেকে (204), ইলিশ (199), এবং (195)
History and Tradition	এই (4809), এবং (4253), করে (3973), ইতিহাস (3396), থেকে (3340), একটি (2227), এর (2119), ছিল (2078), করা (2077), হয়েছে (1964)
Math	গণিত (831), গণিতের (504), এই (501), করে (398), এবং (351), করা (333), বইটি (321), জন্য (299), থেকে (272), সমাধান (271)
Mystery	এক (851), করে (842), এই (821), আর (654), তার (650), থেকে (572), রহস্য (499), কিন্তু (459), সেই (414), আছে (351)
Philosophy	এবং (1150), এই (1116), করে (929), থেকে (649), করা (583), দর্শন (538), তার (536), একটি (497), হয়েছে (473), এর (454)
Politics	এই (1157), এবং (1110), করে (1067), থেকে (734), একটি (634), তিনি (590), করা (583), রাজনৈতিক (561), তার (548), শেখ (535)
Religious	করে (3904), এবং (3412), করা (2988), থেকে (2901), এই (2887), জন্য (2424), এর (2349), তার (2213), আমাদের (1892), আর (1701)
Science Fiction	করে (1208), তার (971), এই (919), এক (898), আর (854), থেকে (817), করতে (528), কিন্তু (481), একটা (455), এবং (443)
Shishu Kishor	করে (1154), তার (908), আর (834), এই (727), এক (597), থেকে (592), কিন্তু (448), একটা (439), কথা (433), আমার (408)
Thriller	এক (1708), করে (1323), তার (1316), এই (1255), আর (1018), থেকে (916), কিন্তু (747), করতে (641), একটা (579), হয়ে (579)

Table 6: Top 10 most frequent Bangla word unigrams for each genre in the BOIGENRE dataset, shown with their raw occurrence counts. The list highlights the strong lexical overlap across narrative genres such as *Contemporary Novel*, *Classic Novel*, and *Shishu Kishor*, where frequent tokens like করে, এই, তার, and এক appear repeatedly across categories. This overlap contributes to cross-genre confusion observed in later analyses (Table 4 and Figure 1).

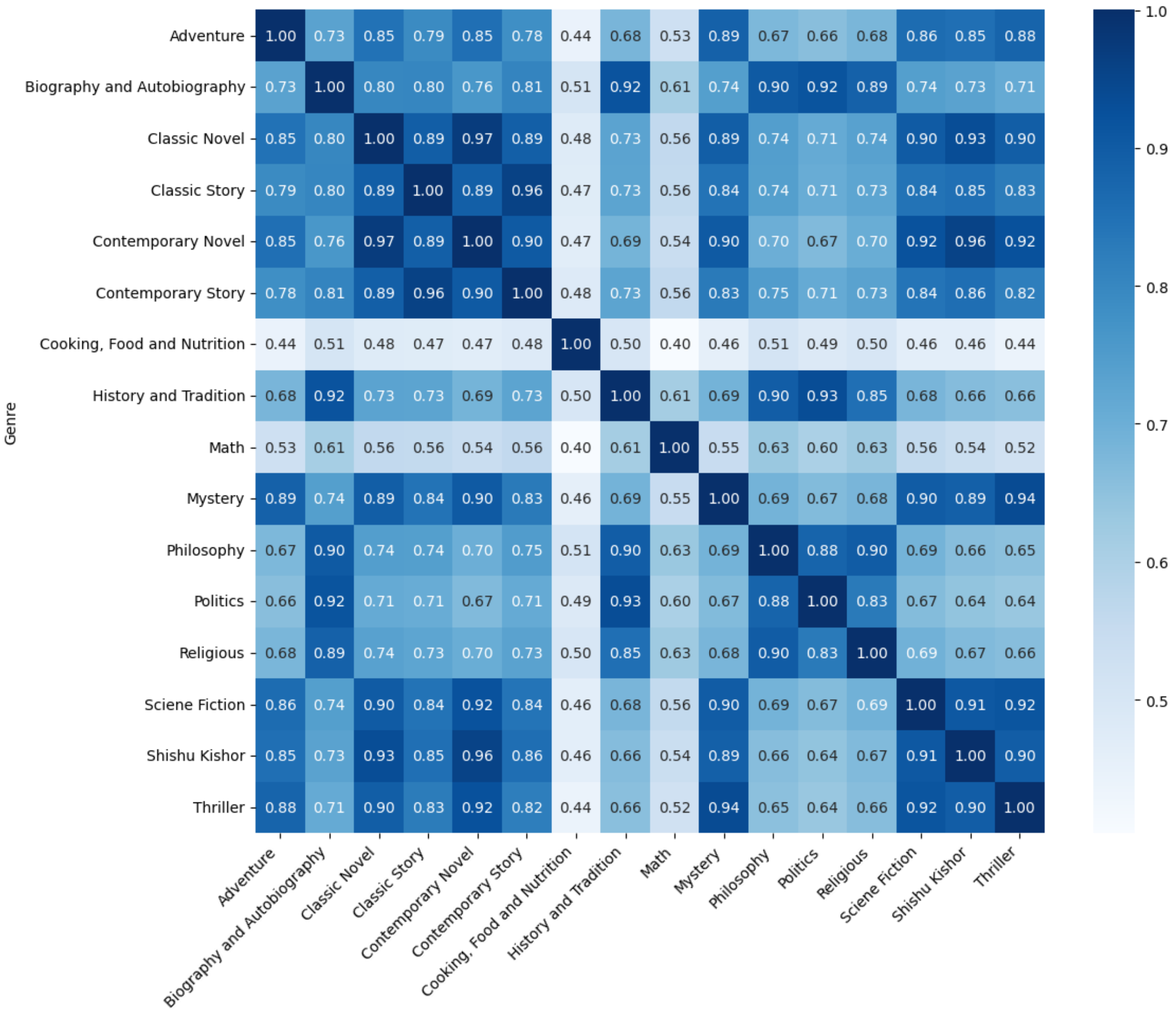
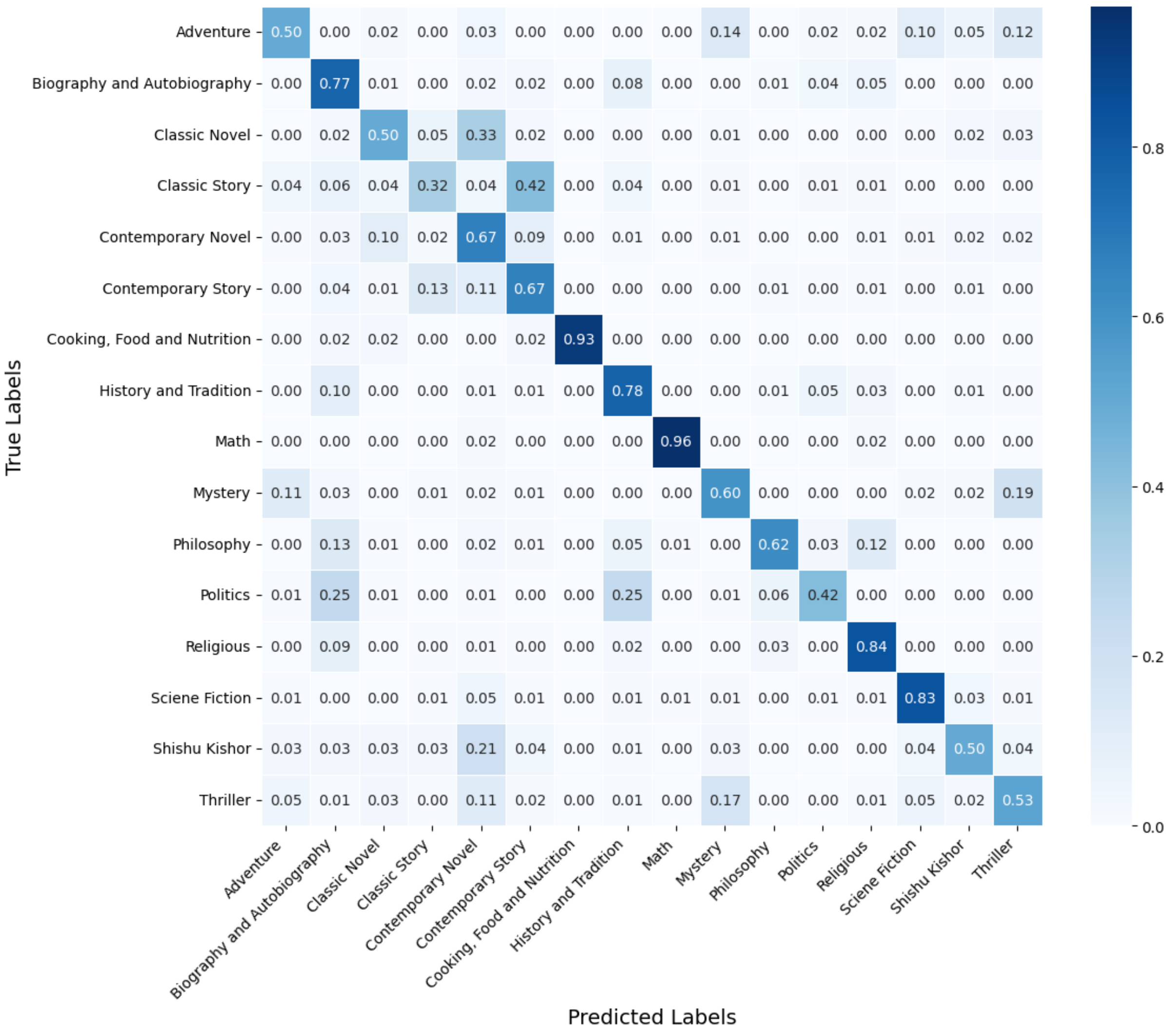


Figure 1: Cosine Similarity of Vocabulary Between Genres (TF-IDF Vocabulary).



	Adventure	Biography and Autobiography	Classic Novel	Classic Story	Contemporary Novel	Contemporary Story	Cooking, Food and Nutrition	History and Tradition	Math	Mystery	Philosophy	Politics	Religious	Sciene Fiction	Shishu Kishor	Thriller
Adventure	0.50	0.00	0.02	0.00	0.03	0.00	0.00	0.00	0.00	0.14	0.00	0.02	0.02	0.10	0.05	0.12
Biography and Autobiography	0.00	0.77	0.01	0.00	0.02	0.02	0.00	0.08	0.00	0.00	0.01	0.04	0.05	0.00	0.00	0.00
Classic Novel	0.00	0.02	0.50	0.05	0.33	0.02	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.02	0.03
Classic Story	0.04	0.06	0.04	0.32	0.04	0.42	0.00	0.04	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00
Contemporary Novel	0.00	0.03	0.10	0.02	0.67	0.09	0.00	0.01	0.00	0.01	0.00	0.00	0.01	0.01	0.02	0.02
Contemporary Story	0.00	0.04	0.01	0.13	0.11	0.67	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.00
Cooking, Food and Nutrition	0.00	0.02	0.02	0.00	0.00	0.02	0.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
History and Tradition	0.00	0.10	0.00	0.00	0.01	0.01	0.00	0.78	0.00	0.00	0.01	0.05	0.03	0.00	0.01	0.00
Math	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.02	0.00	0.00	0.00
Mystery	0.11	0.03	0.00	0.01	0.02	0.01	0.00	0.00	0.00	0.60	0.00	0.00	0.00	0.02	0.02	0.19
Philosophy	0.00	0.13	0.01	0.00	0.02	0.01	0.00	0.05	0.01	0.00	0.62	0.03	0.12	0.00	0.00	0.00
Politics	0.01	0.25	0.01	0.00	0.01	0.00	0.00	0.25	0.00	0.01	0.06	0.42	0.00	0.00	0.00	0.00
Religious	0.00	0.09	0.00	0.00	0.01	0.00	0.00	0.02	0.00	0.00	0.03	0.00	0.84	0.00	0.00	0.00
Sciene Fiction	0.01	0.00	0.00	0.01	0.05	0.01	0.00	0.01	0.01	0.01	0.00	0.01	0.01	0.83	0.03	0.01
Shishu Kishor	0.03	0.03	0.03	0.03	0.21	0.04	0.00	0.01	0.00	0.03	0.00	0.00	0.00	0.04	0.50	0.04
Thriller	0.05	0.01	0.03	0.00	0.11	0.02	0.00	0.01	0.00	0.17	0.00	0.00	0.01	0.05	0.02	0.53

Figure 2: Normalized confusion matrix for BanglaBERT on the BOIGENRE dataset. The diagonal values indicate correctly predicted proportions per genre, while off-diagonal values represent misclassifications between closely related categories.

ChakmaBridge: A Five-Way Parallel Corpus for Navigating the Script Divide in an Endangered Language

Md. Abdur Rahman[1] **Md. Tofael Ahmed Bhuiyan**[1]
Abdul Kadar Muhammad Masum[1]
[1]Department of Computer Science and Engineering, Southeast University
Dhaka, Bangladesh
abdurrahman.etc@gmail.com, tofael1104@gmail.com, akmmasum@seu.edu.bd

Abstract

The advancement of NLP technologies for low-resource and endangered languages is critically hindered by the scarcity of high-quality, parallel corpora. This is particularly true for languages like Chakma, which also faces the challenge of prevalent non-standard, romanized script usage in digital communication. To address this, we introduce ChakmaBridge, the first five-way parallel corpus for Chakma, containing 807 sentences aligned across English, Standard Bangla, Bengali-script Chakma, Romanized Bangla, and Romanized Chakma. Our dataset is created by augmenting the MELD corpus with LLM-generated romanizations that are rigorously validated by native speakers. We establish robust machine translation baselines across six diverse language and script pairs. Our experiments reveal that a multilingual training approach, combining English and Bangla as source languages, yields a dramatic performance increase, achieving a BLEU score of 0.5228 for Chakma translation, a 124% relative improvement over the best bilingual model. We release ChakmaBridge to facilitate research in low-resource MT and aid in the digital preservation of this endangered language.

1 Introduction

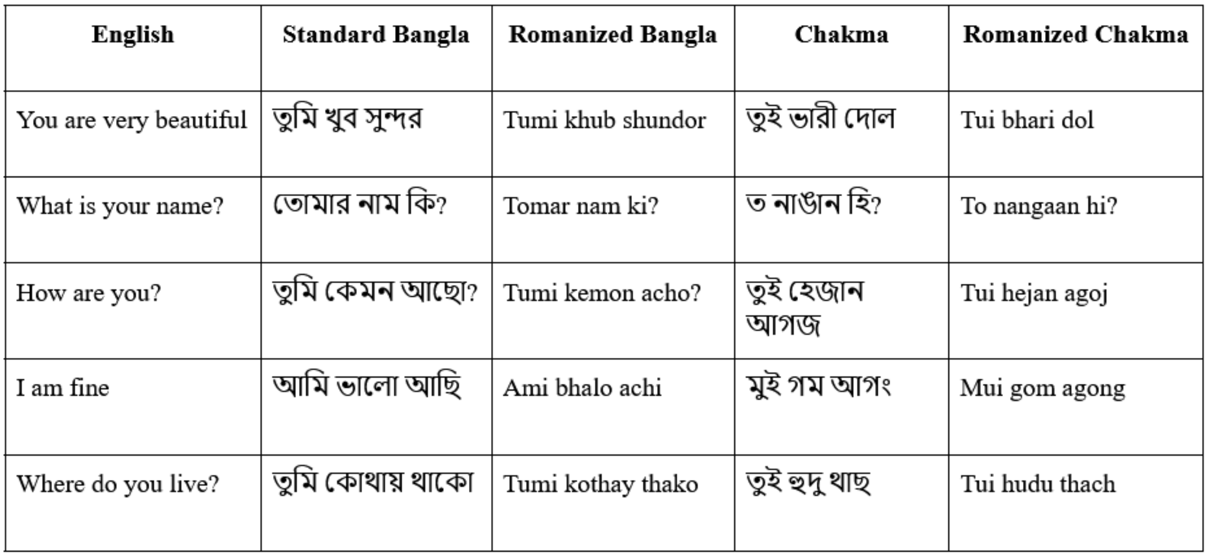
While Large Language Models (LLMs) have driven remarkable progress in Natural Language Processing (NLP), these advancements have not been shared equally across the globe's linguistic landscape (Joshi et al., 2020). Low-resource languages, which are spoken by millions but lack extensive digital corpora, are often excluded from the benefits of modern NLP. This paper addresses this gap by focusing on two such languages from the Indian subcontinent: Bangla and Chakma.

Bangla (Bengali), an Indo-Aryan language with over 270 million native speakers worldwide, is one of the most spoken languages, yet it remains significantly under-resourced in the digital domain compared to languages like English. The Chakma language, also a member of the Indo-Aryan family, is spoken by approximately 800,000 people across communities in Bangladesh, India, and Myanmar (Chakma et al., 2024; Bangladesh Bureau of Statistics, 2023). Despite its significant speaker base, Chakma is classified as "Definitely Endangered" (Saikia and Haokip, 2023), facing immense pressure from dominant regional languages and a critical scarcity of digital resources needed for its preservation and revitalization (Chakma and Maitrot, 2016).

The challenge of digital inclusion is compounded by the widespread use of non-standard scripts in online communication. A prevalent form of text for both Bangla and Chakma on social media and messaging platforms is a romanized version, where the Latin alphabet is used to phonetically represent native words (Moosa et al., 2023). This practice, driven by the ubiquity of QWERTY keyboards, creates a vast but unstructured data source. While this informal text is a valuable linguistic resource, its potential remains largely untapped due to the lack of standardized, parallel datasets that connect it to its native-script counterparts (Roark et al., 2020).

While some foundational computational work exists for Chakma, such as models for character recognition (Podder et al., 2023) and speech identification (Pratap et al., 2024), research in machine translation (MT) remains nascent. Previous studies have highlighted the potential of pre-trained models for Chakma translation but have also underscored the limitations imposed by fragmented and incomplete data (Chakma et al., 2024). we introduce ChakmaBridge, a new, comprehensive parallel dataset for the Chakma language. By releasing this meticulously curated and validated dataset, we aim to empower researchers to develop MT systems that can handle both official scripts and the informal romanized text common in daily dig-

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 259–265
December 23, 2025 ©2025 Association for Computational Linguistics



English	Standard Bangla	Romanized Bangla	Chakma	Romanized Chakma
You are very beautiful	তুমি খুব সুন্দর	Tumi khub shundor	তুই ভারী দোল	Tui bhari dol
What is your name?	তোমার নাম কি?	Tomar nam ki?	ত নাঙান হি?	To nangaan hi?
How are you?	তুমি কেমন আছো?	Tumi kemon acho?	তুই হেজান আগজ	Tui hejan agoj
I am fine	আমি ভালো আছি	Ami bhalo achi	মুই গম আগং	Mui gom agong
Where do you live?	তুমি কোথায় থাকো	Tumi kothay thako	তুই হুদু থাছ	Tui hudu thach

Figure 1: Sample entries from the final five-way parallel ChakmaBridge dataset, illustrating the alignment across English, Standard Bangla, Romanized Bangla, Bengali-script Chakma, and Romanized Chakma.

ital communication. This work seeks to promote linguistic diversity, support the preservation of an endangered cultural heritage, and foster greater digital inclusion for the Chakma-speaking community. Our dataset and the code for our baseline experiments are publicly available on GitHub[1]. Our contributions are two-fold:

1. We introduce ChakmaBridge, the first five-way parallel corpus for Chakma, augmenting the 807 sentences from the MELD dataset (Mahi et al., 2025) with novel, LLM-generated and human-validated columns for `Romanized Bangla` and `Romanized Chakma`.

2. We establish robust translation baselines across six different language and script modalities, demonstrating the dataset's utility and providing a crucial starting point for future research in low-resource MT.

2 Dataset Creation

The creation of our dataset was a multi-stage process designed to build upon an existing resource, augment it with high-quality, semi-automatically generated data, and validate its integrity through a rigorous human-in-the-loop protocol. This section details our data collection and augmentation pipeline, the validation procedure, a statistical analysis of the final corpus, and the experimental setup for establishing baseline translation models.

2.1 Data Collection and Augmentation

Our work commences with the MELD (Multilingual Ethnic Language Dataset), a valuable resource for low-resource languages of Bangladesh (Mahi et al., 2025). The MELD corpus was compiled through structured interviews with native speakers. Crucially, for languages like Chakma that have their own script (Ajhā Pāṭh), the data was phonetically represented using the Bengali script, a common practice for digital communication among the language community. From this resource, we isolated the complete parallel corpus of 807 sentences where Chakma was available, forming a foundational dataset with three aligned columns: `English`, `Standard Bangla`, and `Chakma` (represented in Bengali script).

The primary contribution of our work is the augmentation of this corpus with romanized parallels, addressing the widespread use of Latin script for informal digital communication. To generate the new columns, `Romanized Bangla` and `Romanized Chakma`, we employed a state-of-the-art Large Language Model (LLM), Gemini 2.5 Pro (Comanici et al., 2025). The model was provided with the `Standard Bangla` text as input to produce the `Romanized Bangla` output, and similarly, the Bengali-script `Chakma` text was used to generate the `Romanized Chakma` column. This semi-automated process resulted in ChakmaBridge, a new five-column parallel dataset designed to facilitate research in transliteration, normalization, and multilingual MT between different scripts.

[1] https://github.com/borhanitrash/ChakmaBridge

2.2 Validation Protocol

Given the generative nature of the romanization process, we implemented a rigorous, human-centric validation protocol to ensure the quality and authenticity of the LLM-generated text. The complete set of 807 generated sentence pairs underwent an initial quality assurance review. This process was carried out by a team of three: two senior-year undergraduate students in Computer Science and Engineering, and a professor with a PhD in Electronic Human Resource Management. Their role was to ensure the LLM's output accurately corresponded to the source text, verifying the transliteration alignment rather than the semantic content. Any transliteration errors or outputs that were clearly inconsistent with the input were identified and corrected during this review.

Following this initial pass, we performed a quantitative assessment to measure the alignment between the LLM's output and natural human transliterations. We randomly sampled 20 sentences from the dataset. For the `Romanized Bangla` column, the corresponding `Standard Bangla` sentences were provided to two native Bengali speakers (both undergraduate engineering students) to produce their own romanized versions. For the `Romanized Chakma` column, the Bengali-script `Chakma` sentences were given to two native Chakma speakers (an undergraduate engineering student and a registered nurse). The LLM's romanizations were then systematically compared against the human-generated versions using a suite of standard evaluation metrics: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and the BERTScore F1 (Zhang et al., 2019) to evaluate semantic similarity. The results of this validation are presented in Table 1, demonstrating a high degree of correlation. Further examples illustrating the natural variations between the LLM and native speaker romanizations for both Bangla and Chakma can be found in Appendix A (Figures 2 and 3).

Lang.	Comparison	BLEU	METEOR	BERTScore
Bangla	LLM vs. Native 1	0.7188	0.8837	0.9751
	LLM vs. Native 2	0.6474	0.8308	0.9608
Chakma	LLM vs. Native 1	0.5962	0.7203	0.9391
	LLM vs. Native 2	0.5143	0.7034	0.9009

Table 1: Validation Scores: LLM vs Two Native Speakers (Bangla & Chakma)

2.3 Dataset Analysis and Statistics

The final ChakmaBridge dataset contains 807 parallel sentences. A sample from the corpus, showcasing the five-way alignment across all languages and scripts, is presented in Figure 1. We partitioned the data into training (564 sentences), validation (81 sentences), and test (162 sentences) splits, ensuring a standard 70/10/20 distribution. Table 2 provides a detailed breakdown of the dataset's composition, including the number of tokens and vocabulary size for each language across the splits.

Further statistical properties of the corpus are detailed in Table 3. This analysis highlights key linguistic characteristics; for instance, `Romanized Bangla` has a higher mean character length (34.08) than its Bengali-script counterpart, `Standard Bangla` (30.81), reflecting phonetic spelling conventions. Furthermore, the vocabulary size for Bengali-script `Chakma` (1501 unique words) is the largest in the corpus, underscoring the lexical diversity captured.

2.4 Experimental Setup for Baselines

To demonstrate the utility of our dataset and establish robust benchmarks for future research, we conducted a series of machine translation experiments. We utilized two powerful, publicly available multilingual models: mBART-50 (Li et al., 2021) and NLLB-200 (Costa-Jussà et al., 2022). For all experiments, the models were fine-tuned for 25 epochs using a learning rate of 5e-5, a batch size of 8, and gradient accumulation over 2 steps. Performance was evaluated using BLEU, METEOR, and BERTScore F1. We designed six distinct translation tasks to evaluate performance across various language and script modalities: (1) English to Chakma, (2) Standard Bangla to Chakma, (3) English to Romanized Chakma, (4) Standard Bangla to Romanized Chakma, (5) Romanized Bangla to Romanized Chakma, and (6) a multilingual task of English + Standard Bangla to Chakma. In these tasks, the ‘Chakma‘ target refers to its Bengali script representation. These experiments serve as comprehensive baselines for future work on this low-resource language pair.

3 Results and Discussion

To establish robust baselines and demonstrate the utility of ChakmaBridge, we evaluated the performance of two powerful multilingual models, mBART-50 and NLLB-200, across the six trans-

Language	Number of Sentences				Total Tokens				Vocab Size			
	Train	Valid	Test	Total	Train	Valid	Test	Total	Train	Valid	Test	Total
English	564	81	162	807	3334	480	996	4810	903	238	380	1139
Standard Bangla	564	81	162	807	2989	434	901	4324	1002	239	413	1273
Romanized Bangla	564	81	162	807	2995	435	905	4335	1008	237	413	1275
Chakma	564	81	162	807	3127	457	932	4516	1190	272	467	1501
Romanized Chakma	564	81	162	807	3123	455	933	4511	1136	267	453	1431

Table 2: Data Statistics per Language: Sentences, Tokens, Vocabulary (Train, Val, Test)

Column	Mean Char. Length	Max Char. Length	Min Char. Length	Mean Word Count	Max Word Count	Min Word Count	Unique Word Count	Unique Sent. Count
English	30.65	102	7	5.96	16	2	1192	775
Standard Bangla	30.81	83	9	5.36	12	2	1273	789
Romanized Bangla	34.08	95	10	5.37	12	2	1316	784
Chakma	28.62	83	6	5.60	14	2	1501	801
Romanized Chakma	31.39	94	7	5.59	14	2	1484	803
Total Dataset	**31.11**	**102**	**6**	**5.58**	**16**	**2**	**5973**	**3952**

Table 3: Detailed character-level and word-level statistics for each column in the dataset.

lation tasks outlined in Section 2.4. The complete results, measured by BLEU, METEOR, and BERTScore F1, are presented in Table 4.

Overall Performance and Model Comparison. Across most tasks, NLLB-200 generally outperforms mBART-50. For instance, in the Standard Bangla → Chakma task, NLLB-200 achieves a BLEU score of 0.2330, surpassing mBART-50's score of 0.2016. This trend suggests that NLLB-200's broader pre-training on a more diverse set of languages, including those from the Indo-Aryan family, provides a better foundation for fine-tuning on low-resource pairs like Bangla-Chakma. While the BLEU and METEOR scores are modest, which is expected for such a low-resource scenario, the high BERTScore F1 values (consistently above 0.80) indicate that the models are successful in capturing the semantic content of the translations, even when n-gram overlap is limited.

Impact of Source Language. Comparing the performance of English vs. Standard Bangla as the source language reveals an interesting pattern. For translation into Bengali-script Chakma, using Standard Bangla as the source yields consistently better results than using English. NLLB-200 achieves a BLEU score of 0.2330 from Bangla, compared to 0.1975 from English. This is likely due to the significant lexical and syntactic similarities between Bangla and Chakma, both being Eastern Indo-Aryan languages. The shared vocabulary and grammatical structures provide a stronger transfer learning signal.

Challenges of Script Transliteratio. The experiments involving romanized targets highlight the added complexity of script conversion. The task of translating from Standard Bangla to Romanized Chakma proves challenging, with NLLB-200 achieving a BLEU score of 0.1867. This task requires the model to perform both semantic translation (Bangla to Chakma) and phonetic transliteration (Bengali script to Latin script) simultaneously. Interestingly, the direct translation from Romanized Bangla to Romanized Chakma (BLEU of 0.2057 with NLLB-200) performs better, suggesting that when the source and target share the same script, the model can focus more effectively on the linguistic translation task.

The Power of Multilingual Training. The most significant finding is the dramatic performance improvement in the multilingual training setting. By jointly training on both English and Standard Bangla source sentences, the NLLB-200 model's performance skyrockets, achieving a BLEU score of 0.5228, a METEOR score of 0.6486, and a BERTScore F1 of 0.9136. This represents a relative improvement of over 124% in BLEU score compared to the next best individual task (Standard Bangla → Chakma). This substantial gain likely stems from several factors. First, the larger and more diverse training data provides a powerful regularization effect, preventing the model from overfitting on the small ChakmaBridge training set. Second, by being exposed to the syntactically distinct English (SVO) alongside the similar Bangla (SOV), the model is forced to learn more abstract and robust cross-lingual representations rather than relying on surface-level pattern matching between the two related Indo-Aryan languages. This encourages the development of a more generalized translation capability.

Translation Direction	Model	BLEU	METEOR	BERTScore F1
English → Chakma	mBART-50	0.1788	0.3449	0.8340
	NLLB-200	0.1975	0.3710	0.8474
Standard Bangla → Chakma	mBART-50	0.2016	0.3640	0.8451
	NLLB-200	0.2330	0.4042	0.8578
English → Romanized Chakma	mBART-50	0.1836	0.3217	0.8794
	NLLB-200	0.1489	0.3038	0.7911
Standard Bangla → Romanized Chakma	mBART-50	0.1664	0.3365	0.7916
	NLLB-200	0.1867	0.3505	0.8857
Romanized Bangla → Romanized Chakma	mBART-50	0.1980	0.3914	0.8247
	NLLB-200	0.2057	0.3817	0.8246
Multilingual (English + Standard Bangla) → Chakma	mBART-50	0.2744	0.4608	0.8646
	NLLB-200	**0.5228**	**0.6486**	**0.9136**

Table 4: Translation Performance Across Directions and Models (BLEU, METEOR, BERTScore F1)

Qualitative Error Analysis. To better understand the models' performance beyond quantitative metrics, we conducted a qualitative analysis of common errors in the output of the best-performing multilingual model (NLLB-200). We identified three primary categories of errors: (1) Lexical Choice Errors, where the model selects a semantically related but contextually incorrect word (e.g., translating "house" as "building"). (2) Grammatical Errors, primarily involving incorrect postpositions or verb conjugations, which are common challenges in morphologically rich languages like Chakma. (3) Omissions, where the model fails to translate a specific noun or adjective from the source sentence, leading to a loss of detail. These errors suggest that while the model effectively captures the overall meaning, as indicated by the high BERTScore, it still struggles with fine-grained lexical and syntactic details, a typical challenge when fine-tuning on a small dataset. Future work could explore data augmentation or constrained decoding techniques to mitigate these specific error types.

4 Conclusion

In this work, we introduced ChakmaBridge, a novel five-way parallel corpus designed to advance NLP research for the low-resource, endangered Chakma language. By augmenting the foundational MELD dataset with human-validated, LLM-generated romanized parallels for both Chakma and Bangla, we have created a unique resource that reflects modern digital communication practices. Our comprehensive baseline experiments demonstrate the dataset's utility for a range of machine translation tasks across different scripts. Notably, our findings reveal that a multilingual training approach, combining both English and the linguistically closer Bangla, dramatically improves translation quality into Chakma, boosting performance by over 124% in BLEU score. We release ChakmaBridge to the community with the hope that it will spur further research, aid in the development of practical translation tools, and contribute to the digital preservation of the Chakma language and its rich cultural heritage. Future efforts should build upon this by focusing on dataset expansion, exploring data augmentation strategies, and incorporating direct support for the native Chakma script.

Limitations

While ChakmaBridge represents a significant step forward, we acknowledge several limitations that offer clear avenues for future research. First, the primary limitation is the modest size of the corpus. At only 807 sentences, it is insufficient for training robust neural models from scratch and restricts the generalizability of our findings. While our multilingual approach serves as a mitigation strategy by leveraging high-resource data, future work should explore targeted data augmentation techniques like back-translation to artificially expand the training set. Second, our romanization methodology has inherent constraints. The LLM-generated text, while validated, is likely more standardized and cleaner than the noisy, orthographically diverse "in-the-wild" romanization found in user-generated content. This could limit the real-world applicability of models trained on this data. Furthermore, our validation protocol, based on a small sample of 20 sentences, provides a preliminary quality check but is not extensive enough to offer definitive statistical confidence. Finally, our work is scoped to Chakma as represented in the Bengali script. A key omission is the lack of evaluation or direct support for the native Chakma (Ajhā Pāṭh) script. Developing resources and models capable of processing and generating the native script is a critical and necessary next step for the comprehensive digital revitalization of the language.

Acknowledgments

This work was supported by Southeast University, Bangladesh. We would like to thank Writu Raj Chakma and Akhi Chakma for their valuable assistance in validating the dataset.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Bangladesh Bureau of Statistics. 2023. Table a-1.4 ethnic population by group and sex. in: *Statistics*, 2023. Based on Bangladesh Bureau of Statistics 2021 data.

Aunabil Chakma, Aditya Chakma, Soham Khisa, Chumui Tripura, Masum Hasan, and Rifat Shahriyar. 2024. Chakmanmt: A low-resource machine translation on chakma language. *arXiv preprint arXiv:2410.10219*.

Nikhil Chakma and Mathilde Maitrot. 2016. How ethnic minorities became poor and stay poor in bangladesh: A qualitative enquiry. *EEP/Shiree, July*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, and 1 others. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Xian Li, Changhan Wang, Yun Tang, Chau Tran, Yuqing Tang, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2021. Multilingual speech translation from efficient finetuning of pretrained models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 827–838, Online. Association for Computational Linguistics.

Mehraj Hossain Mahi, Anzir Rahman Khan, Mobashsher Hasan Anik, Sheak Rashed Haider Noori, Arif Mahmud, and Mayen Uddin Mojumdar. 2025. Meld: A multilingual ethnic dataset of chakma, garo, and marma in bengali script with english and standard bengali translation. *Data in Brief*, page 111745.

Ibraheem Muhammad Moosa, Mahmud Elahi Akhter, and Ashfia Binte Habib. 2023. Does transliteration help multilingual language modeling? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 670–685, Dubrovnik, Croatia. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Kanchon Kanti Podder, Ludmila Emdad Khan, Jyoti Chakma, Muhammad EH Chowdhury, Proma Dutta, Khan Md Anwarus Salam, Amith Khandakar, Mohamed Arselene Ayari, Bikash Kumar Bhawmick, SM Arafin Islam, and 1 others. 2023. Self-chakmanet: A deep learning framework for indigenous language learning using handwritten characters. *Egyptian Informatics Journal*, 24(4):100413.

Vineel Pratap, Andros Tjandra, Bowen Shi, Paden Tomasello, Arun Babu, Sayani Kundu, Ali Elkahky, Zhaoheng Ni, Apoorv Vyas, Maryam Fazel-Zarandi, and 1 others. 2024. Scaling speech technology to 1,000+ languages. *Journal of Machine Learning Research*, 25(97):1–52.

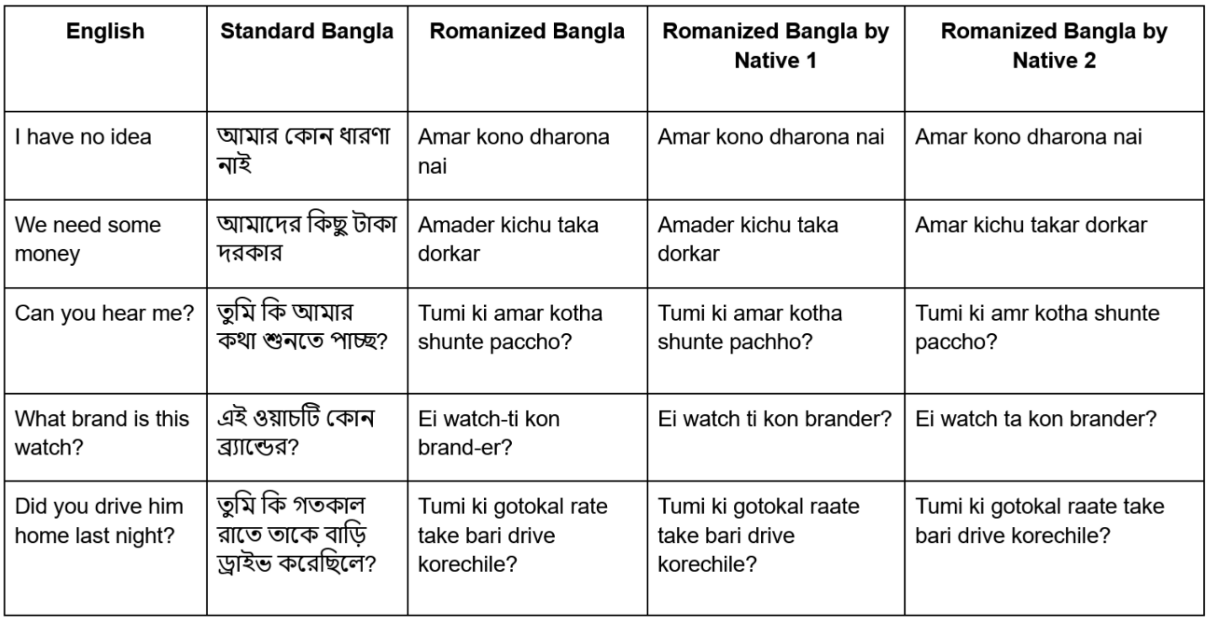
Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020. Processing South Asian languages written in the Latin script: the Dakshina dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2413–2423, Marseille, France. European Language Resources Association.

Jonali Saikia and Mary Kim Haokip. 2023. Language endangerment with special refrence to chakma. *Journal of English Language and Literature*, 10(3).

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

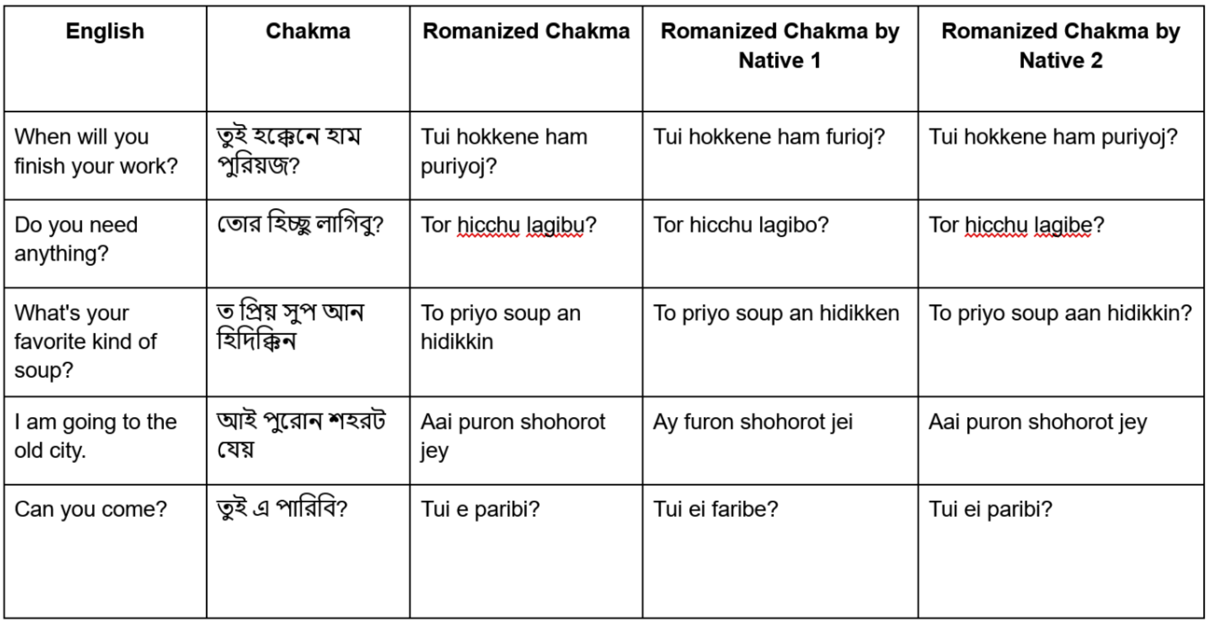
A Dataset and Validation Samples

This appendix provides visual examples from the ChakmaBridge dataset. Figure 2 and Figure 3 present examples from our validation protocol, illustrating the comparison between the LLM-generated romanizations and those produced by native speakers.



English	Standard Bangla	Romanized Bangla	Romanized Bangla by Native 1	Romanized Bangla by Native 2
I have no idea	আমার কোন ধারণা নাই	Amar kono dharona nai	Amar kono dharona nai	Amar kono dharona nai
We need some money	আমাদের কিছু টাকা দরকার	Amader kichu taka dorkar	Amader kichu taka dorkar	Amar kichu takar dorkar
Can you hear me?	তুমি কি আমার কথা শুনতে পাচ্ছ?	Tumi ki amar kotha shunte paccho?	Tumi ki amar kotha shunte pachho?	Tumi ki amr kotha shunte paccho?
What brand is this watch?	এই ওয়াচটি কোন ব্র্যান্ডের?	Ei watch-ti kon brand-er?	Ei watch ti kon brander?	Ei watch ta kon brander?
Did you drive him home last night?	তুমি কি গতকাল রাতে তাকে বাড়ি ড্রাইভ করেছিলে?	Tumi ki gotokal rate take bari drive korechile?	Tumi ki gotokal raate take bari drive korechile?	Tumi ki gotokal raate take bari drive korechile?

Figure 2: Validation samples for Romanized Bangla, comparing the LLM-generated output with versions from two native speakers. The table highlights the natural diversity in phonetic romanization; for instance, the sound ’ch’ is variably rendered as ’cch’ (paccho) or ’chh’ (pachho). Such nuances demonstrate the challenge of standardizing romanized text and the importance of our validation process.



English	Chakma	Romanized Chakma	Romanized Chakma by Native 1	Romanized Chakma by Native 2
When will you finish your work?	তুই হক্কেনে হাম পুরিয়জ?	Tui hokkene ham puriyoj?	Tui hokkene ham furioj?	Tui hokkene ham puriyoj?
Do you need anything?	তোর হিচ্ছু লাগিবু?	Tor hicchu lagibu?	Tor hicchu lagibo?	Tor hicchu lagibe?
What's your favorite kind of soup?	ত প্রিয় সুপ আন হিদিক্কিন	To priyo soup an hidikkin	To priyo soup an hidikken	To priyo soup aan hidikkin?
I am going to the old city.	আই পুরোন শহরট যেয়	Aai puron shohorot jey	Ay furon shohorot jei	Aai puron shohorot jey
Can you come?	তুই এ পারিবি?	Tui e paribi?	Tui ei faribe?	Tui ei paribi?

Figure 3: Validation samples for Romanized Chakma. This figure demonstrates key phonetic distinctions often found in native Chakma romanization that may be missed by automated systems. For example, the alternation between ’p’ and ’f’ sounds (e.g., puriyoj vs. furioj and paribi vs. faribe) reflects a common phonological variation. Capturing these subtle, real-world differences is crucial for developing robust transliteration and translation models.

A Comparative Analysis of Retrieval-Augmented Generation Techniques for Bengali Standard-to-Dialect Machine Translation Using LLMs

K. M. Jubair Sami, Dipto Sumit, Ariyan Hossain, Farig Sadeque
Department of Computer Science and Engineering
BRAC University, Dhaka, Bangladesh
{km.jubair.sami, dipto.sumit}@g.bracu.ac.bd
{ariyan.hossain, farig.sadeque}@bracu.ac.bd

Abstract

Translating from a standard language to its regional dialects is a significant NLP challenge due to scarce data and linguistic variation, a problem prominent in the Bengali language. This paper proposes and compares two novel RAG pipelines for standard-to-dialectal Bengali translation. The first, a Transcript-Based Pipeline, uses large dialect sentence contexts from audio transcripts. The second, a more effective Standardized Sentence-Pairs Pipeline, utilizes structured local_dialect:standard_bengali sentence pairs. We evaluated both pipelines across six Bengali dialects and multiple LLMs using BLEU, ChrF, WER, and BERTScore. Our findings show that the sentence-pair pipeline consistently outperforms the transcript-based one, reducing Word Error Rate (WER) from 76% to 55% for the Chittagong dialect. Critically, this RAG approach enables smaller models (e.g., Llama-3.1-8B) to outperform much larger models (e.g., GPT-OSS-120B), demonstrating that a well-designed retrieval strategy can be more crucial than model size. This work contributes an effective, fine-tuning-free solution for low-resource dialect translation, offering a practical blueprint for preserving linguistic diversity.

1 Introduction

The Bengali language's diverse and culturally significant regional dialects (Paul et al., 2024; Khandaker et al., 2025; Wasi et al., 2024) are critically underrepresented in machine translation (MT) (Khandaker et al., 2025). While some research translates dialects into standard Bengali (Faria et al., 2023), the reverse task: translating from standard to regional variants, remains a more challenging and largely unexplored problem (Khandaker et al., 2025). This gap is driven by the lack of parallel standard-to-dialect corpora, a common challenge for low-resource languages (Klementiev et al., 2012; Yakhni and Chehab, 2025). Consequently, Large Language Models (LLMs) often fail to capture subtle dialectal nuances without specialized guidance, resulting in inaccurate translations (Yakhni and Chehab, 2025; Kadaoui et al., 2023).

This paper's contributions are as follows:

- We design and evaluate two distinct, fine-tuning-free pipelines for standard-to-dialectal Bengali translation using in-context learning: (1) a Transcript-Based Pipeline that uses dialectal audio transcripts as context for large LLMs, and (2) a Standardized Sentence-Pairs Pipeline that uses standard-dialect sentence pairs for smaller LLMs.
- We systematically compare these approaches across multiple Large Language Models (LLMs) and six underrepresented Bengali dialects: Chittagong, Comilla, Habiganj, Rangpur, Sylhet, and Tangail.
- Our findings identify the optimal strategy for different conditions, providing a practical blueprint for developing Machine Translation (MT) systems for low-resource dialects.

2 Related Works

The task of translating between standard languages and dialects presents significant challenges. To address these, our work is situated at the intersection of existing research on dialect processing and emerging methodological approaches, which we review below.

Bangla Dialect Processing. Research in Bangla dialect processing has largely focused on identification and dialect-to-standard translation, leveraging the Vashantor dataset (Faria et al., 2023). This corpus, covering the Chittagong, Noakhali, Sylhet, Barishal, and Mymensingh dialects, has been used to train fine-tuned models and prompt LLMs for these tasks (Faria et al., 2023; Paul et al., 2024). In contrast, the data-scarce standard-to-dialect direction is less explored. This reverse task was addressed by Khandaker et al. (2025) via fine-tuning neural models on the Vashantor dataset. Our work also tackles this challenge, proposing a fine-tuning-free, retrieval-augmented alternative.

Dialect Translation in Other Languages. Similar challenges exist elsewhere; studies on Arabic explore fine-tuning and prompting for dialect translation (Alabdullah et al., 2025). Research on Lebanese Arabic highlights LLMs' failure to capture cultural nuances without authentic data (Yakhni and Chehab, 2025), underscoring our transcript-based approach and the practice of comparing different methods. (Alabdullah et al., 2025; Liu et al., 2023; Han et al., 2024).

Applying RAG to Dialect Translation. Our pipelines utilize Retrieval-Augmented Generation (RAG), where retrieved dialect sentence pairs or transcript excerpts serve as few-shot in-context examples for an LLM. While RAG is established for question-answering, its application to low-resource dialect translation is an emerging area(Perak et al., 2024; Ndimbo et al., 2025; Kyslyi et al., 2025; Miyagawa, 2025). This RAG-inspired approach mitigates data scarcity and helps preserve culturally specific lexical and pragmatic patterns during translation.

3 Methodology

We compare two strategies for standard-to-dialectal Bangla translation, using two distinct datasets each tailored to a specific pipeline.

3.1 Datasets

3.1.1 Dataset-01: Transcript-Based Dataset (for Pipeline 1)

District	# of data points
Sylhet	7,624
Kishoreganj	2,049
Narail	1,859
Chittagong	1,757
Narsingdi	1,373
Sandwip	1,310
Rangpur	1,298
Tangail	1,271
Habiganj	1,170
Barishal	1,006
Comilla	318
Noakhali	278
Total	21,313

Table 1: Dialect coverage and number of sentences in the transcript-based dataset.

This dataset (Hassan et al., 2025), from transcribed audio of local Bengali dialects, contains long, contextually rich sentences reflecting spoken language. Its broad district coverage captures diverse lexical and syntactic variation, ideal for in-context retrieval by large LLMs.

3.1.2 Dataset-02: Standardized Sentence-Pairs Dataset (for Pipeline 2)

Structured as key-value pairs of local_dialect_sentence:standard_bengali_translation (Hassan et al., 2025), the raw data initially contained many small, fragmented sentences. Our preprocessing attempt to merge them yielded modest improvement in similarity search performance.

District	Before preprocessing	After preprocessing
Chittagong	7,193	7,295
Habiganj	5,375	5,457
Rangpur	4,061	4,140
Kishoreganj	3,653	3,898
Tangail	353	365
Total	20,635	21,155

Table 2: Dialect coverage and number of sentence-pairs in the standardized dataset, shown before and after preprocessing.

3.2 Dataset Preprocessing and Indexing

We developed two distinct preprocessing and indexing pipelines for our retrieval systems to accommodate significant differences between our datasets: the first dataset contains long, formal sentences (mean 38.2 words), while the second has short, conversational fragments (mean 6.9 words), necessitating a more intensive and specialized preprocessing approach.

3.2.1 Pipeline 1: Standard Preprocessing

For Dataset-01, our pipeline focused on robust cleaning and direct embedding. The key steps were:

Text Cleaning: We loaded raw transcriptions, filtered invalid data, and ensured consistent UTF-8 encoding.

Metadata and Quality Metrics: Each entry was augmented with metadata (ID, dialect, length) and quality metrics like word count and text complexity.

Hybrid Indexing: We adopted a hybrid retrieval approach for both semantic and lexical matching:

Dense Index: We generated 768-dimensional embeddings using the l3cube-pune/bengali-sentence-similarity-sbert model (Deode et al., 2023), a model specifically fine-tuned for semantic similarity on Bengali text. These were L2-

normalized and indexed with FAISS[1] (IndexFlatIP) for efficient cosine similarity search (Douze et al., 2024).

Sparse Index: Concurrently, we built a rank_bm25 index for keyword-based sparse retrieval (Robertson and Zaragoza, 2009).

3.2.2 Pipeline 2: Augmented Preprocessing for Short Texts

The shorter sentence pairs in Dataset-02 required a more sophisticated pipeline to enrich contextual information before embedding.

Systematic Text Normalization: We applied a multi-step normalization function including Unicode NFC, standardization of Bengali digits and punctuation, and collapsing repeated whitespace and characters.

Short Fragment Augmentation: To add crucial context to short texts, we tagged sentences with fewer than three tokens as [[SHORT]] and applied content-based tags like [[QUESTION]]. Consecutive short entries from the same dialect were merged into a single, contextually rich record marked [[MERGED]].

Structured Representation: Before embedding, each entry was formatted as: District: {district} | STANDARD: {standard_norm} | LOCAL: {local_norm_tagged}. This structure explicitly provides the model with dialectal, standard, and augmented local information to learn region-specific translation patterns.

Hybrid Indexing: As in the first pipeline, we generated hybrid dense (FAISS) and sparse (BM25) indices from these structured representations to enhance retrieval performance.

The intensive augmentation step was designed to address the fact that shorter sentences in Dataset-02 lack self-contained context. Our merging and tagging strategies artificially created this context, providing a richer signal to the embedding model and mitigating the ambiguity of short utterances.

3.3 Translation Pipelines

3.3.1 Pipeline 1: Transcript-Based Pipeline for Larger LLMs

This pipeline is designed for simplicity and is particularly effective for large, powerful LLMs that have been pre-trained on extensive Bengali data. The workflow is as follows:

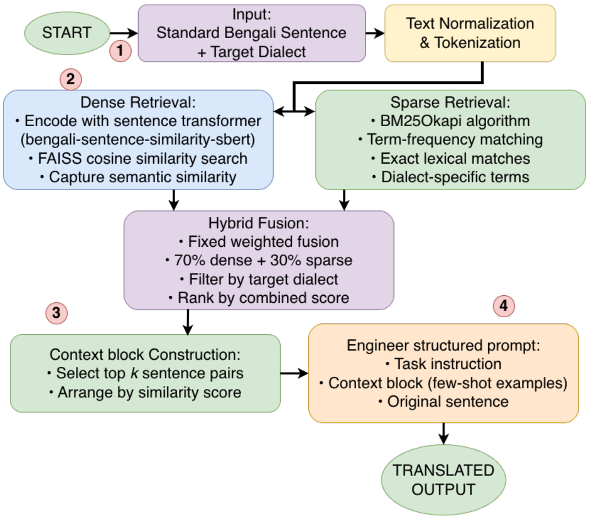

START
1
Input:
Standard Bengali Sentence
+ Target Dialect
Text Normalization
& Tokenization
2
Dense Retrieval:
• Encode with sentence transformer
(bengali-sentence-similarity-sbert)
• FAISS cosine similarity search
• Capture semantic similarity
Sparse Retrieval:
• BM25Okapi algorithm
• Term-frequency matching
• Exact lexical matches
• Dialect-specific terms
Hybrid Fusion:
• Fixed weighted fusion
• 70% dense + 30% sparse
• Filter by target dialect
• Rank by combined score
3
Context block Construction:
• Select top k sentence pairs
• Arrange by similarity score
4
Engineer structured prompt:
• Task instruction
• Context block (few-shot examples)
• Original sentence
TRANSLATED
OUTPUT

Figure 1: Pipeline 1 translation workflow.

Input: A standard Bengali sentence and target dialect are provided by the user. The input sentence then undergoes standard text normalization and tokenization to prepare it for processing.

Hybrid Vector-Based Retrieval: We use a hybrid system to find relevant examples, combining two methods. For **Dense Retrieval**, the same sentence transformer from indexing generates an embedding of the input to find semantically similar sentences via a cosine similarity search on the FAISS index. For **Sparse Retrieval**, a BM25Okapi algorithm performs term-frequency based matching to identify sentences with exact lexical matches and key dialect-specific terms. Finally, a **Hybrid Fusion** combines the scores using a weighted fusion (70% dense, 30% sparse), and the results are then filtered by the target dialect and ranked.

Context Construction & LLM-Based Translation: A few-shot context is constructed by selecting the top n (a user-defined hyperparameter) sentence pairs, ranked by similarity. This context, along with a task instruction and the input sentence, is then formatted into a prompt and fed to an LLM to generate the final translation. A sample prompt is provided in Appendix B.

3.3.2 Pipeline 2: Standardized Sentence-Pairs Pipeline for Smaller LLMs

This pipeline is more complex, designed to maximize retrieval accuracy as Dataset-02 has relatively smaller sentence pairs. Since it retrieves both the local_dialect and standard_bengali sentence pairs, it is also designed for smaller, more efficient LLMs that might not be pre-trained on extensive Bengali data. The workflow is as follows:

[1] https://github.com/facebookresearch/faiss

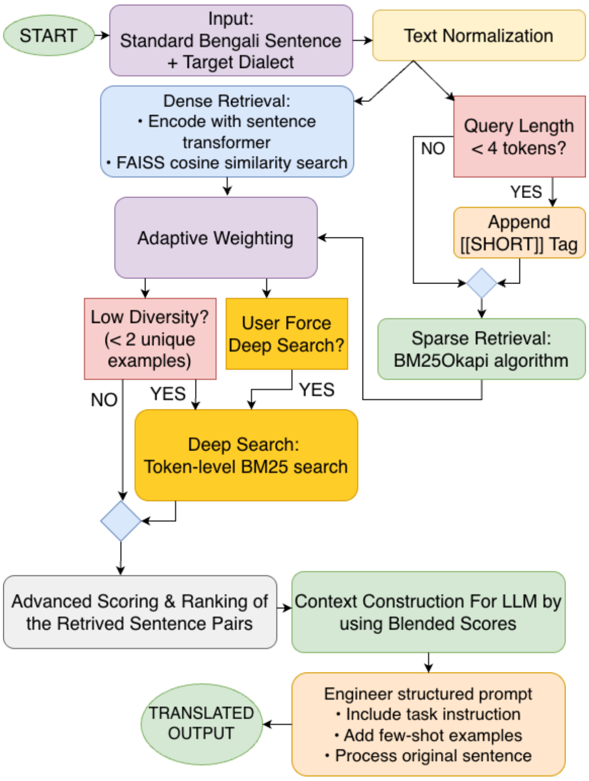

START
Input:
Standard Bengali Sentence
+ Target Dialect
Text Normalization
Dense Retrieval:
• Encode with sentence transformer
• FAISS cosine similarity search
Query Length < 4 tokens?
NO
YES
Append [[SHORT]] Tag
Adaptive Weighting
Sparse Retrieval:
BM25Okapi algorithm
Low Diversity? (< 2 unique examples)
User Force Deep Search?
NO
YES
YES
Deep Search:
Token-level BM25 search
Advanced Scoring & Ranking of the Retrived Sentence Pairs
Context Construction For LLM by using Blended Scores
Engineer structured prompt
• Include task instruction
• Add few-shot examples
• Process original sentence
TRANSLATED OUTPUT

Figure 2: Pipeline 2 translation workflow.

Input and Normalization: The input Standard Bengali sentence undergoes a comprehensive normalization process, which includes Unicode normalization, removal of zero-width characters, punctuation standardization, and numeral conversion. Queries shorter than four tokens are tagged as short.

Hybrid Retrieval with Adaptive Weighting: We identify relevant sentence pairs using a hybrid approach with dynamic weights. For **Dense Retrieval**, the same sentence transformer from indexing encodes the input for a FAISS cosine similarity search to find semantically similar examples. **Sparse Retrieval** uses BM25 for lexical matching. The appended [[SHORT]] tag to the input is to specifically target other short examples in the corpus. The fusion employs **Adaptive Weighting** based on query length: standard queries favor dense retrieval (55/35), while short queries prioritize sparse retrieval (35/55) to better capture lexical matches. Furthermore, the number of candidates retrieved is doubled for both sparse (50 to 200) and dense (50 to 100) searches to cast a wider net for short queries.

Deep Search for Low-Diversity Queries: A "Deep Search" mechanism is initiated either automatically when initial results lack diversity (e.g., fewer than two unique examples) or manually by the user. It runs a BM25 search for each input token, aggregates the scores, and re-weights to favor sparse retrieval.

Advanced Scoring and Ranking: Candidates from the retrieval stages are ranked using a blended score. This final score incorporates the weighted dense and sparse similarity scores, along with several bonuses, including a district matching bonus, significant bonuses for exact and substring matches, and a minor bonus based on character-level similarity.

Context Construction & LLM-Based Translation: A few-shot context is constructed by filtering top n (a user-defined hyperparameter) ranked sentence pairs by the target dialect, and sorting by score. A prompt containing these standard_bengali:local_dialect examples, along with instructions and the input sentence, is then sent to an LLM to generate the final translation. A sample prompt is provided in Appendix B.

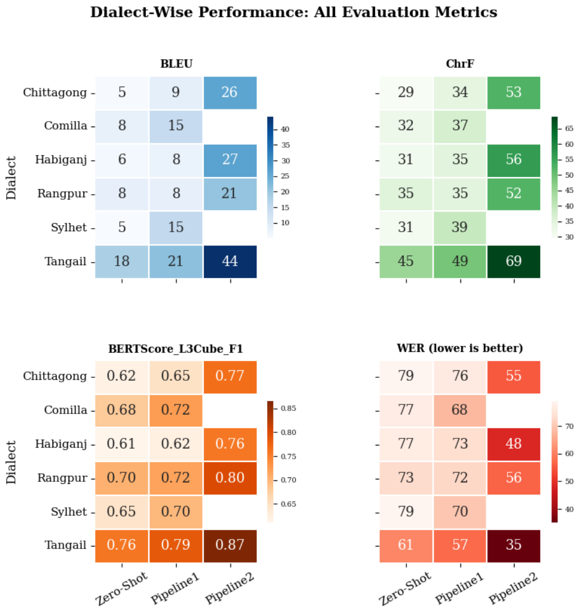
4 Experiments and Results

4.1 Experimental Setup

To investigate the relationship between model characteristics and pipeline design in dialectal translation, we evaluated our pipelines across a diverse set of LLMs, ranging from smaller open-weight models to larger ones, as well as proprietary models. The comparison covered six Bengali dialects: Chittagong, Habiganj, Rangpur, Tangail (present in both datasets), and Comilla and Sylhet (only in Dataset-01). We assessed translation quality using complementary metrics covering lexical overlap (BLEU (Papineni et al., 2002), ChrF (Popović, 2015)), edit distance (WER), and learned semantic similarity (BERTScore F1 (Zhang* et al., 2020)), evaluated on $N = 50$ diverse sentence pairs per dialect, totaling 7,700 data points across all pipeline-dialect combinations. Detailed metric formulations and implementation specifics are provided in Appendix A.

5 Results and Analysis

We evaluated both pipelines across multiple LLMs and six Bengali dialects. Figure 3 presents a comprehensive performance overview, with scores averaged across all LLMs for each pipeline-dialect combination. It is important to note that this averaging can sometimes mask the peak performance of the best models, as lower-performing models can pull down the aggregate score. Complete performance tables showing individual LLM results for all dialects are provided in Appendix E. Nevertheless, Pipeline 2 consistently outperforms Pipeline 1, a difference largely attributable to its superior data structure and preprocessing. This performance gap is also qualitatively evident in the prompts themselves. As illustrated in Appendix B using a consistent example sentence, the structured few-shot pairs in Pipeline 2 produce highly accurate translations, whereas Pipeline 1 only partially captures dialectal nuances and the Zero-Shot baseline fails entirely. Our quantitative analysis also shows that dialectal proximity to Standard Bengali strongly correlates with translation quality, and well-designed RAG pipelines enable smaller models to compete with larger ones. Detailed model-wise comparisons are shown in Appendix C. A comparison with Khandaker et al. (2025)'s fine-tuned models is provided in Appendix D.


Dialect-Wise Performance: All Evaluation Metrics

BLEU

Dialect	Zero-Shot	Pipeline1	Pipeline2
Chittagong	5	9	26
Comilla	8	15	
Habiganj	6	8	27
Rangpur	8	8	21
Sylhet	5	15	
Tangail	18	21	44

ChrF

Dialect	Zero-Shot	Pipeline1	Pipeline2
Chittagong	29	34	53
Comilla	32	37	
Habiganj	31	35	56
Rangpur	35	35	52
Sylhet	31	39	
Tangail	45	49	69

BERTScore_L3Cube_F1

Dialect	Zero-Shot	Pipeline1	Pipeline2
Chittagong	0.62	0.65	0.77
Comilla	0.68	0.72	
Habiganj	0.61	0.62	0.76
Rangpur	0.70	0.72	0.80
Sylhet	0.65	0.70	
Tangail	0.76	0.79	0.87

WER (lower is better)

Dialect	Zero-Shot	Pipeline1	Pipeline2
Chittagong	79	76	55
Comilla	77	68	
Habiganj	77	73	48
Rangpur	73	72	56
Sylhet	79	70	
Tangail	61	57	35

Figure 3: Dialect-wise performance comparison across zero-shot, Pipeline 1, and Pipeline 2 settings, with scores averaged across all LLMs. The hierarchy is clear: Pipeline 2 > Pipeline 1 > Zero-shot.

5.1 Pipeline Comparison

As shown in Figure 3, Pipeline 2 systematically outperforms Pipeline 1 across all shared dialects (e.g., Chittagong: BLEU 9→26, WER 76%→55%). This stems from Dataset-02's explicit local_dialect:standard_bengali pairs providing ideal few-shot context versus Dataset-01's raw transcripts, plus significantly higher number of data points (Chittagong: 7,295 vs. 1,757 examples) with advanced preprocessing for short fragments.

5.2 Linguistic Proximity Dominates

Dialectal similarity to Standard Bengali is the strongest performance predictor. Tangail achieves the highest scores (BLEU=44, WER=35) with only 365 examples, while divergent dialects like Chittagong (WER=55) and Sylhet/Comilla (WER=70/68) require both abundant data and intensive preprocessing. Intermediate dialects (Habiganj, Rangpur: WER=48/56) show moderate divergence can be mitigated via Pipeline 2. Critically, this linguistic proximity advantage persists even in zero-shot scenarios: Tangail achieves BLEU=18 and WER=61 without any dialectal examples, outperforming divergent dialects Chittagong (BLEU=5, WER=79) and Sylhet (BLEU=5, WER=79) by 3.4-3.6× in BLEU scores, confirming that inherent linguistic similarity to Standard Bengali remains the dominant factor regardless of learning paradigm.

5.3 LLM Performance

Zero-shot translation consistently fails (BLEU=5-12, WER=67-84%). Pipeline 2 enables dramatic gains: Gemma-3-27B improves from WER=76.62% to 36.70%, achieving best overall performance (BLEU=45.06). Critically, smaller models like Llama-3.1-8B (WER=51.18) outperform much larger models like GPT-OSS-120B (WER=52.65), demonstrating retrieval quality can compensate for model capacity (Appendix C.1).

6 Conclusion and Future Work

We proposed two RAG-based pipelines for standard-to-dialectal Bengali translation. Pipeline 2 (Standardized Sentence-Pairs) proves most effective, enabling smaller models to outperform higher-parameter counterparts by converting an intractable zero-shot task into a manageable few-shot problem. While linguistic proximity to Standard Bengali strongly correlates with performance, our fine-tuning-free approach provides a practical blueprint for low-resource dialect translation. This work is ongoing: we are actively expanding Dataset-02, developing a more optimized version of Pipeline 2, and investigating fine-tuning-based approaches alongside retrieval-augmented methods.

Limitations and Challenges

Despite the promising results, this study is subject to several limitations and faced inherent challenges that warrant discussion.

- **Data Availability and Quality:** The primary challenge remains the scarcity of high-quality, parallel corpora for Bengali dialects. While our pipelines aimed to mitigate this, their performance is still fundamentally constrained by the volume and cleanliness of the underlying datasets. The datasets used contained inconsistencies and noise inherent to transcribed spoken language, which could impact retrieval accuracy.

- **Limited Dialectal Coverage:** Our evaluation was confined to six Bengali dialects. Given the vast number of dialects spoken across Bangladesh and West Bengal, our findings may not be generalizable to all linguistic variants, especially those with more pronounced structural differences from Standard Bengali.

- **Evaluation Constraints:** Our evaluation was constrained by limited time, computational resources, and the availability of human annotators. Consequently, we utilized a curated test set of $N = 50$ diverse sentence pairs per dialect. Across all combinations of pipelines and dialects, this amounted to a total of 7,700 data points. While curated for diversity, this sample size is a limitation; more robust results would require a larger test set. A second major limitation is our exclusive reliance on automated metrics. These metrics fail to capture critical nuances of dialectal appropriateness, fluency, and cultural context, which can only be assessed through human evaluation.

- **Absence of a Production-Ready Baseline:** A direct comparison with fine-tuned models on the exact same standard-to-dialect task was not performed within this study's scope. While Khandaker et al. (2025) explored fine-tuning using smaller, neural models, a head-to-head comparison would be needed to precisely quantify the trade-offs between RAG with LLMs and fine-tuned smaller models.

Ethical Considerations

Developing technology for low-resource dialects carries significant ethical responsibilities. While this work aims to support linguistic diversity, it is crucial to consider the potential impacts.

- **Preservation vs. Misrepresentation:** The goal is to preserve and promote dialectal use. However, inaccurate or culturally insensitive translations generated by automated systems risk misrepresenting the language and its speakers. There is a danger of propagating stereotypes or producing nonsensical text that could undermine the perceived value of the dialect.

- **Data Sovereignty and Consent:** The datasets used in this research were drawn from existing public collections. Future data collection efforts must prioritize ethical practices, including obtaining informed consent from native speakers, ensuring fair compensation for their linguistic expertise, and respecting community ownership of the data.

- **Inadvertent Standardization:** The creation of translation tools, by its nature, involves a degree of standardization. There is a risk that such tools could inadvertently promote a single, computationally convenient version of a dialect, thereby eroding the rich, organic micro-variations that exist within dialect communities. Engagement with linguists and community members is vital to mitigate this risk.

- **Usage of AI Tools:** We acknowledge the use of AI-based writing assistants in the preparation of this paper for improving grammar and style. The core ideas, experimental design, and analysis were conducted entirely by the authors.

References

Abdullah Alabdullah, Lifeng Han, and Chenghua Lin. 2025. Advancing dialectal arabic to modern standard arabic machine translation. *Preprint*, arXiv:2507.20301.

Samruddhi Deode, Janhavi Gadre, Aditi Kajale, Ananya Joshi, and Raviraj Joshi. 2023. L3Cube-IndicSBERT: A simple approach for learning cross-lingual sentence representations using multilingual BERT. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 154–163, Hong Kong, China. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Fatema Tuj Johora Faria, Mukaffi Bin Moin, Ahmed Al Wase, Mehidi Ahmmed, Md. Rabius Sani, and Tashreef Muhammad. 2023. Vashantor: A large-scale multilingual benchmark dataset for automated translation of bangla regional dialects to bangla language. *Preprint*, arXiv:2311.11142.

Lifeng Han, Serge Gladkoff, Gleb Erofeev, Irina Sorokina, Betty Galiano, and Goran Nenadic. 2024. Neural machine translation of clinical text: an empirical investigation into multilingual pre-trained language models and transfer-learning. *Frontiers in Digital Health*, 6:1211564.

Md. Rezuwan Hassan, Azmol Hossain, Kanij Fatema, Rubayet Sabbir Faruque, Tanmoy Shome, Ruwad Naswan, Trina Chakraborty, Md. Foriduzzaman Zihad, Tawsif Tashwar Dipto, Nazia Tasnim, Nazmuddoha Ansary, Md. Mehedi Hasan Shawon, Ahmed Imtiaz Humayun, Md. Golam Rabiul Alam, Farig Sadeque, and Asif Sushmit. 2025. Regspeech12: A regional corpus of bengali spontaneous speech across dialects. *Preprint*, arXiv:2510.24096.

Karima Kadaoui, Samar M. Magdy, Abdul Waheed, Md Tawkat Islam Khondaker, Ahmed Oumar El-Shangiti, El Moatez Billah Nagoudi, and Muhammad Abdul-Mageed. 2023. TARJAMAT: Evaluation of bard and ChatGPT on machine translation of ten Arabic varieties. In *Proceedings of ArabicNLP 2023*, pages 52–75, Singapore (Hybrid). Association for Computational Linguistics.

Md. Arafat Alam Khandaker, Ziyan Shirin Raha, Bidyarthi Paul, and Tashreef Muhammad. 2025. Bridging dialects: Translating standard bangla to regional variants using neural models. In *Proceedings of the 27th International Conference on Computer and Information Technology (ICCIT)*, Cox's Bazar, Bangladesh. Preprint available via arXiv.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 130–140, Avignon, France. Association for Computational Linguistics.

Roman Kyslyi, Yuliia Maksymiuk, and Ihor Pysmennyi. 2025. Vuyko mistral: Adapting LLMs for low-resource dialectal translation. In *Proceedings of the Fourth Ukrainian Natural Language Processing Workshop (UNLP 2025)*, pages 86–95, Vienna, Austria (online). Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

So Miyagawa. 2025. RAG-enhanced neural machine translation of Ancient Egyptian text: A case study of THOTH AI. In *Proceedings of the 5th International Conference on Natural Language Processing for Digital Humanities*, pages 33–40, Albuquerque, USA. Association for Computational Linguistics.

Edmund V. Ndimbo, Qin Luo, Gimo C. Fernando, Xu Yang, and Bang Wang. 2025. Leveraging retrieval-augmented generation for swahili language conversation systems. *Applied Sciences*, 15(2).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Bidyarthi Paul, Faika Fairuj Preotee, Shuvashis Sarker, and Tashreef Muhammad. 2024. Improving bangla regional dialect detection using bert, llms, and xai. In *2024 IEEE International Conference on Computing, Applications and Systems (COMPAS)*, pages 1–6.

Benedikt Perak, Slobodan Beliga, and Ana Meštrović. 2024. Incorporating dialect understanding into LLM using RAG and prompt engineering techniques for causal commonsense reasoning. In *Proceedings of the Eleventh Workshop on NLP for Similar Languages, Varieties, and Dialects (VarDial 2024)*, pages 220–229, Mexico City, Mexico. Association for Computational Linguistics.

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, Lisbon, Portugal. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389.

Azmine Toushik Wasi, Taki Rafi, and Dong-Kyu Chae. 2024. Diaframe: A framework for understanding bengali dialects in human-ai collaborative creative writing spaces. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 268–274.

Silvana Yakhni and Ali Chehab. 2025. Can LLMs translate cultural nuance in dialects? a case study on Lebanese Arabic. In *Proceedings of the 1st Workshop on NLP for Languages Using Arabic Script*, pages 114–135, Abu Dhabi, UAE. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Evaluation Metrics: Detailed Formulations

To provide a comprehensive assessment of translation quality, we report a set of complementary metrics that cover lexical overlap, edit distance, and learned semantic similarity. All metrics were evaluated on $N = 50$ sentence pairs per dialect, covering a wide range of Bengali lexical diversity.

1. **Corpus-level overlap metrics (BLEU and ChrF)**: We report BLEU (Papineni et al., 2002) and ChrF (Popović, 2015) as corpus-level overlap metrics. Both metrics are computed at the corpus level by aggregating their underlying counts (for example, n-gram matches and candidate/reference lengths) across all sentences before applying the final scoring formula. Aggregating counts prior to final computation preserves the correct statistical behavior of these metrics and avoids inflation that can occur when averaging sentence-level scores on small test sets. Formally, let M denote either BLEU or ChrF, and let Numerator$_i$ and Denominator$_i$ be the per-sentence internal counts used by M. The corpus score is calculated by aggregating those counts across all sentences and then applying the metric's scoring function:

$$M_{\text{corpus}} = M\Big(\textstyle\sum_{i=1}^{N} \text{Numerator}_i,\ \sum_{i=1}^{N} \text{Denominator}_i\Big). \tag{1}$$

2. **Edit-distance metric (WER)**: Word Error Rate (WER) measures the minimum number of substitutions (S), deletions (D) and insertions (I) needed to convert a hypothesis into a reference, normalized by the reference length. For a robust corpus-level estimate we weight each sentence's WER by its reference word count (RefWC$_i$) and compute the length-weighted average:

$$\text{WER}_{\text{corpus}} = \frac{\sum_{i=1}^{N} \text{WER}_i \times \text{RefWC}_i}{\sum_{i=1}^{N} \text{RefWC}_i}, \tag{2}$$

where WER$_i$ is the sentence-level WER for sentence i.

3. **Learned semantic similarity (BERTScore F1)**: BERTScore F1 (Zhang* et al., 2020) computes a soft token-level alignment using contextual embeddings (we used the L3Cube Bengali variant to generate embeddings to calculate BERTScores) and produces a continuous similarity score per sentence. As a learned metric, BERTScore is far more robust than n-gram methods (BLEU) at capturing semantic equivalence, which is particularly valuable for evaluating low-resource languages like Bengali where lexical variation is common. Because BERTScore is designed as a sentence-level metric, we report the final corpus-level BERTScore as the arithmetic mean of the N sentence scores:

$$M_{\text{corpus}} = \frac{1}{N} \sum_{i=1}^{N} M_i, \tag{3}$$

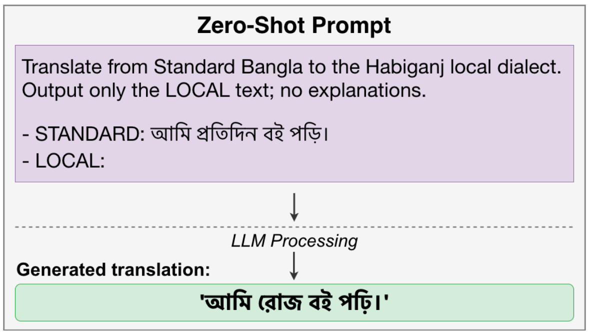
where M_i is the BERTScore F1 of sentence i.

4. **Implementation details**: All metrics were computed using standard, publicly available implementations with default settings unless otherwise noted. In particular, BLEU and ChrF were computed using corpus-level aggregation (not averaged segment BLEU/ChrF), WER was computed using a standard minimum-edit-distance alignment and length-weighted aggregation, and BERTScore F1 was computed at the sentence level and averaged across the evaluation set.

B Illustrative Prompt Examples and Translation Quality

This section presents a comparative analysis of prompts from our three experimental setups: Zero-Shot, Pipeline 1 (Transcript-Based), and Pipeline 2 (Standardized Sentence-Pairs). To clearly demonstrate the impact of each prompting strategy, we use the same standard Bengali input sentence, target dialect, and LLM in all examples. The resulting translations highlight a clear progression in quality: the Zero-Shot approach completely fails to capture dialectal features, Pipeline 1 partially captures them, and Pipeline 2 produces the most accurate and fluent dialectal output.

B.1 Zero-Shot: Prompt Example



Zero-Shot Prompt

Translate from Standard Bangla to the Habiganj local dialect.
Output only the LOCAL text; no explanations.

- STANDARD: আমি প্রতিদিন বই পড়ি।
- LOCAL:

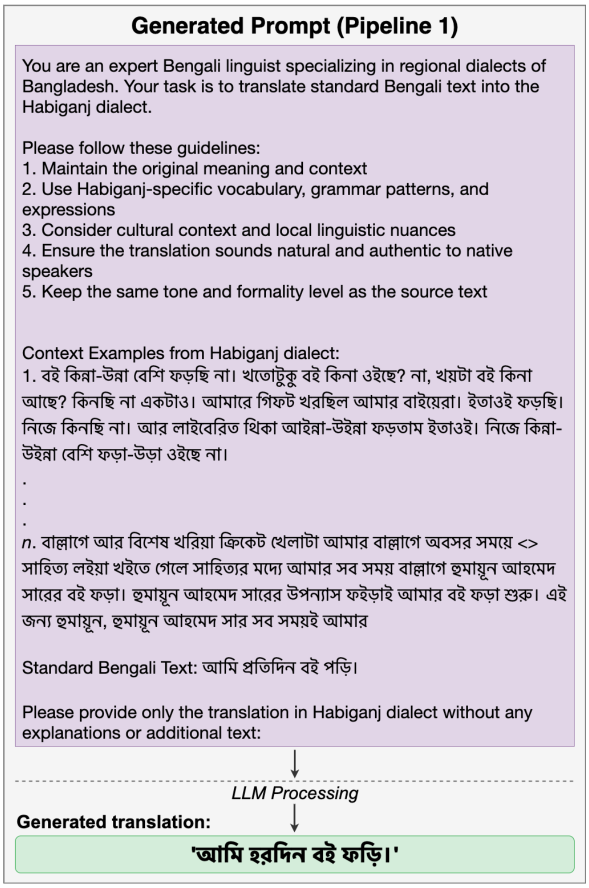
LLM Processing

Generated translation:

'আমি রোজ বই পড়ি।'

Figure 4: A sample prompt that used while translating a sentence in Zero-Shot scenarios.

B.2 Pipeline 1: Transcript-Based Prompt Example



Generated Prompt (Pipeline 1)

You are an expert Bengali linguist specializing in regional dialects of Bangladesh. Your task is to translate standard Bengali text into the Habiganj dialect.

Please follow these guidelines:
1. Maintain the original meaning and context
2. Use Habiganj-specific vocabulary, grammar patterns, and expressions
3. Consider cultural context and local linguistic nuances
4. Ensure the translation sounds natural and authentic to native speakers
5. Keep the same tone and formality level as the source text

Context Examples from Habiganj dialect:
1. বই কিন্না-উন্না বেশি ফড়ছি না। খতোটুকু বই কিনা ওইছে? না, খয়টা বই কিনা আছে? কিনছি না একটাও। আমারে গিফট খরছিল আমার বাইয়েরা। ইতাওই ফড়ছি। নিজে কিনছি না। আর লাইবেরিত থিকা আইন্না-উইন্না ফড়তাম ইতাওই। নিজে কিন্না-উইন্না বেশি ফড়া-উড়া ওইছে না।
.
.
.
n. বাল্লাগে আর বিশেষ খরিয়া ক্রিকেট খেলাটা আমার বাল্লাগে অবসর সময়ে <> সাহিত্য লইয়া খইতে গেলে সাহিত্যর মদ্যে আমার সব সময় বাল্লাগে হুমায়ূন আহমেদ সারের বই ফড়া। হুমায়ূন আহমেদ সারের উপন্যাস ফইড়াই আমার বই ফড়া শুরু। এই জন্য হুমায়ূন, হুমায়ূন আহমেদ সার সব সময়ই আমার

Standard Bengali Text: আমি প্রতিদিন বই পড়ি।

Please provide only the translation in Habiganj dialect without any explanations or additional text:

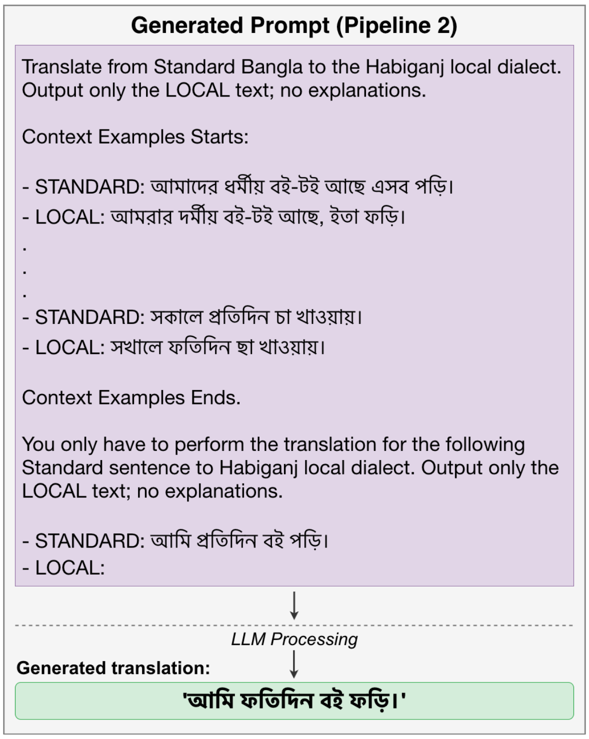
LLM Processing

Generated translation:

'আমি হরদিন বই ফড়ি।'

Figure 5: A sample prompt that generated while translating a sentence using Pipeline 1 (Transcript-Based Pipeline). The prompt contains up to n retrieved context examples.

B.3 Pipeline 2: Standardized Sentence-Pairs Prompt Example



Generated Prompt (Pipeline 2)

Translate from Standard Bangla to the Habiganj local dialect.
Output only the LOCAL text; no explanations.

Context Examples Starts:

- STANDARD: আমাদের ধর্মীয় বই-টই আছে এসব পড়ি।
- LOCAL: আমরার দর্মীয় বই-টই আছে, ইতা ফড়ি।
.
.
.
- STANDARD: সকালে প্রতিদিন চা খাওয়ায়।
- LOCAL: সখালে ফতিদিন ছা খাওয়ায়।

Context Examples Ends.

You only have to perform the translation for the following Standard sentence to Habiganj local dialect. Output only the LOCAL text; no explanations.

- STANDARD: আমি প্রতিদিন বই পড়ি।
- LOCAL:

LLM Processing

Generated translation:

'আমি ফতিদিন বই ফড়ি।'

Figure 6: A sample prompt that generated while translating a sentence using Pipeline 2 (Standardized Sentence-Pairs Pipeline). The prompt includes up to n retrieved standard_bengali:local_dialect sentence pairs as few-shot examples.

C Detailed Model Performance Analysis

This section provides comprehensive performance breakdowns for all evaluated LLMs across zero-shot, Pipeline 1, and Pipeline 2 conditions. All values are reported as mean ± standard deviation across all dialects evaluated.

C.1 Model Comparison Across Conditions

C.2 Zero-Shot Performance

The zero-shot baseline establishes that standard-to-dialect Bengali translation is a challenging task requiring contextual examples. Performance is uniformly low across all models:

- **Best performing models**: Gemini-2.5Fash (BLEU=12.06±6.53, WER=68.47±10.01) and GPT-OSS-20B (BLEU=11.72±9.65, WER=67.37±11.90) show marginally better results but remain far from acceptable translation quality.
- **Smaller models**: Models like Gemma-3-12B (BLEU=5.21±3.14, WER=83.56±5.04)

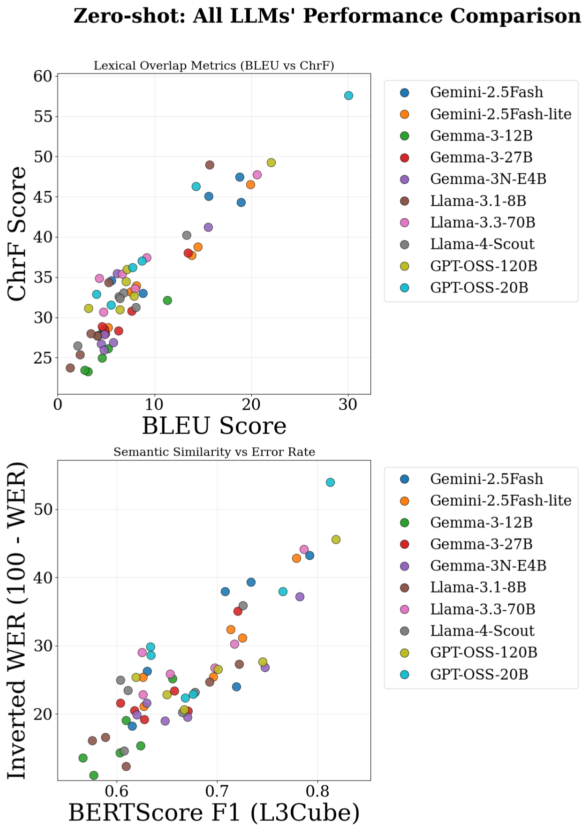


Figure 7: Performance comparison of different LLMs in zero-shot scenario. All models show uniformly poor performance (BLEU 5-12, WER 67-84%), confirming the necessity of RAG-based approaches for this task.

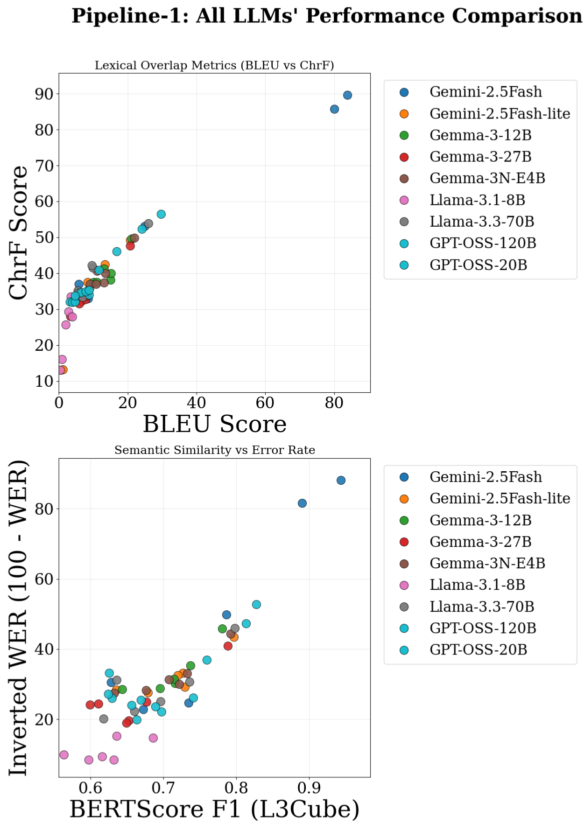


Figure 8: Performance comparison of different LLMs using Pipeline 1 (transcript-based). Gemini-2.5Fash shows notably superior performance (BLEU=34.87, WER=50.39), suggesting stronger ability to infer dialectal patterns from less structured context.

and Llama-3.1-8B (BLEU=5.35±5.26, WER=82.39±7.16) struggle significantly without contextual guidance.

- **Key insight**: Even large 70B+ parameter models fail to produce quality dialectal translations zero-shot, with BLEU scores consistently below 12 and WER above 67%. This confirms the task's inherent difficulty and data scarcity.

C.3 Pipeline 1 Performance

Pipeline 1 uses transcript-based context with longer, more descriptive dialectal sentences. Performance improvements over zero-shot are modest for most models, with one notable exception:

- **Gemini-2.5Fash (outlier)**: Achieves BLEU=34.87±37.15 and WER=50.39±29.05, dramatically outperforming all other models in this pipeline. The high standard deviations suggest strong performance on some dialects but inconsistency across others.
- **Other models**: Most models show minimal improvement over zero-shot. For example, Gemma-3-12B improves to BLEU=14.27±3.91 (from 5.21), while GPT-OSS-120B remains at BLEU=9.04±7.69 (barely changed from 8.96).
- **Llama-3.1-8B failure**: This model performs worse than zero-shot (BLEU=2.22±1.39, WER=88.96±3.10), suggesting the transcript-based context format may confuse smaller models lacking sufficient Bengali pretraining.

C.4 Pipeline 2 Performance

Pipeline 2 provides structured local_dialect:standard_bengali sentence pairs, resulting in dramatic and consistent improvements across all models:

- **Top tier models**: Gemma-3-27B leads with BLEU=45.06±15.67 and WER=36.70±11.41, followed closely by Gemini-2.5Fash-lite (BLEU=31.80±13.59, WER=47.14±11.75) and Gemini-2.5Fash (BLEU=30.62±12.51, WER=47.88±12.62).

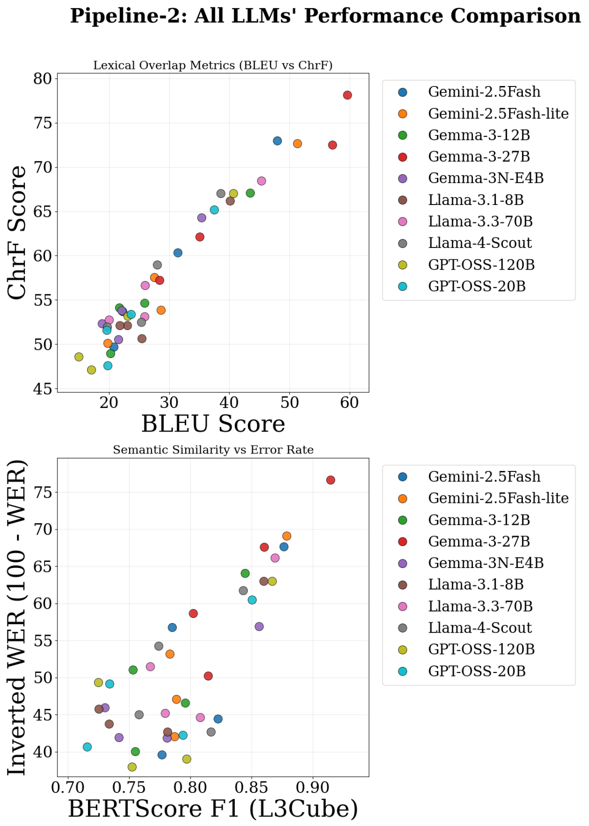


Figure 9: Performance comparison of different LLMs using Pipeline 2 (standardized sentence-pairs). Gemma-3-27B achieves the best overall results (BLEU=45.06, WER=36.70). The superior pipeline narrows the performance gap between models of different sizes.

- **Mid-tier performance**: Models like Llama-3.3-70B (WER=48.12±10.02), Llama-4-Scout (WER=49.07±8.77), and Gemma-3-12B (WER=49.56±10.16) cluster in the 48-50% WER range, demonstrating solid translation quality.

- **Smaller models competitive**: Llama-3.1-8B (WER=51.18±9.56) and Gemma-3N-E4B (WER=53.33±7.11) become viable options, performing comparably to much larger models like GPT-OSS-120B (WER=52.65±11.64).

- **Performance convergence**: Standard deviations decrease substantially compared to Pipeline 1, indicating more consistent performance across dialects with this approach.

C.5 Key Takeaways

1. **RAG is essential**: The zero-shot baseline confirms that LLMs lack intrinsic dialectal translation knowledge, regardless of size.

2. **Data structure matters more than volume**: Pipeline 2's explicit sentence pairs outperform Pipeline 1's longer transcripts, even with similar data volumes.

3. **Smaller models become viable**: With proper context, 8B-12B parameter models can outperform 70B-120B models, democratizing dialectal translation.

4. **Model selection depends on pipeline**: Gemini-2.5Fash excels in Pipeline 1, while Gemma-3-27B dominates Pipeline 2, suggesting different architectural strengths.

D Comparison with Fine-Tuned Baseline

To contextualize our RAG-based approach, we compare our pipelines against the fine-tuned models from Khandaker et al. (2025), who pioneered supervised standard-to-dialect Bengali translation. For this comparison, presented in Table 3, we've taken the best result from our LLM-Pipeline-Dialect combinations, focusing on the Chittagong and Sylhet dialects as these are the only ones where both studies use Word Error Rate (WER) as a common metric. It is important to note the methodological differences: Khandaker et al. (2025) utilized supervised fine-tuning of BanglaT5 on parallel corpora, which demands significant training data and resources. In contrast, our approach is fine-tuning-free, relying on retrieval-based in-context learning.

Dialect	Khandaker et al.	Pipeline 1 (Best)	Pipeline 2 (Best)
Chittagong	70.66	71.16	**32.37**
Sylhet	60.64	**18.37**	-

Table 3: WER (%) comparison between Khandaker et al.'s fine-tuned BanglaT5 model and our RAG-based pipelines for shared dialects.

E Complete Performance Tables

This section presents comprehensive performance tables for all three experimental setups: Pipeline 1 (Transcript-Based), Pipeline 2 (Standardized Sentence-Pairs), and Zero-Shot. Each table shows results for all evaluated LLMs across all dialects, with the best performance for each dialect-metric combination highlighted in bold.

E.1 Pipeline 1: Complete Results

E.2 Pipeline 2: Complete Results

E.3 Zero-Shot: Complete Results

Dialect	**Model**	**BLEU**	**ChrF**	**BERTScore F1**	**WER ↓**
Chittagong	gemini-2.5-flash	8.53	32.89	0.6722	77.18
Chittagong	gemini-2.5-flash-lite	8.90	35.86	0.6790	72.41
Chittagong	**gemma-3-12b-it**	**14.95**	**38.20**	**0.6955**	**71.16**
Chittagong	gemma-3-27b-it	6.47	32.20	0.6112	75.52
Chittagong	gemma-3n-e4b-it	13.12	37.34	0.6764	71.78
Chittagong	llama-3.1-8b-instant	2.72	29.25	0.5974	91.49
Chittagong	llama-3.3-70b-versatile	5.56	35.12	0.6178	79.88
Chittagong	openaigpt-oss-120b	8.74	33.97	0.6692	74.48
Chittagong	openaigpt-oss-20b	8.81	35.23	0.6244	72.82
Comilla	**gemini-2.5-flash**	**83.83**	**89.63**	**0.9433**	**11.79**
Comilla	gemini-2.5-flash-lite	0.63	13.05	0.7270	66.81
Comilla	gemma-3-12b-it	15.15	39.96	0.7374	64.63
Comilla	gemma-3-27b-it	3.33	28.04	0.6528	80.42
Comilla	gemma-3n-e4b-it	13.54	39.99	0.7326	66.95
Comilla	llama-3.1-8b-instant	1.98	25.65	0.6324	91.58
Comilla	llama-3.3-70b-versatile	6.91	33.29	0.6960	74.95
Comilla	openaigpt-oss-120b	3.23	32.10	0.6976	77.89
Comilla	openaigpt-oss-20b	4.71	32.01	0.6890	76.42
Habiganj	gemini-2.5-flash	5.82	37.03	0.6284	69.49
Habiganj	gemini-2.5-flash-lite	8.32	37.46	0.6354	71.61
Habiganj	gemma-3-12b-it	10.17	37.48	**0.6435**	71.40
Habiganj	gemma-3-27b-it	7.65	32.68	0.5992	75.85
Habiganj	gemma-3n-e4b-it	8.97	36.95	0.6329	72.46
Habiganj	llama-3.1-8b-instant	0.93	16.11	0.5636	90.04
Habiganj	llama-3.3-70b-versatile	9.84	**41.55**	0.6360	68.86
Habiganj	openaigpt-oss-120b	6.46	34.63	0.6292	73.94
Habiganj	**openaigpt-oss-20b**	**11.59**	40.92	0.6256	**66.74**
Rangpur	gemini-2.5-flash	6.06	34.68	0.7344	75.32
Rangpur	gemini-2.5-flash-lite	1.10	13.24	0.7293	70.82
Rangpur	gemma-3-12b-it	11.10	37.46	0.7158	69.74
Rangpur	gemma-3-27b-it	8.02	33.73	0.6770	75.11
Rangpur	gemma-3n-e4b-it	10.76	36.93	0.7218	69.96
Rangpur	llama-3.1-8b-instant	3.51	33.52	0.6359	84.76
Rangpur	llama-3.3-70b-versatile	9.59	42.24	0.7359	69.31
Rangpur	openaigpt-oss-120b	7.87	34.96	0.7411	73.82
Rangpur	**openaigpt-oss-20b**	**16.71**	**46.12**	**0.7595**	**63.09**
Sylhet	**gemini-2.5-flash**	**80.03**	**85.78**	**0.8904**	**18.37**
Sylhet	gemini-2.5-flash-lite	13.40	42.42	0.7197	67.43
Sylhet	gemma-3-12b-it	13.12	41.24	0.7146	68.48
Sylhet	gemma-3-27b-it	5.97	31.53	0.6491	81.00
Sylhet	gemma-3n-e4b-it	11.11	40.60	0.7075	68.68
Sylhet	llama-3.1-8b-instant	0.35	13.01	0.6158	90.61
Sylhet	llama-3.3-70b-versatile	5.59	34.57	0.6605	77.66
Sylhet	openaigpt-oss-120b	3.81	31.89	0.6635	80.17
Sylhet	openaigpt-oss-20b	4.71	33.73	0.6566	75.99
Tangail	gemini-2.5-flash	24.96	53.06	0.7866	50.21
Tangail	gemini-2.5-flash-lite	20.74	49.16	0.7970	56.51
Tangail	gemma-3-12b-it	21.12	49.53	0.7807	54.20
Tangail	gemma-3-27b-it	20.69	47.70	0.7883	59.03
Tangail	gemma-3n-e4b-it	21.90	49.80	0.7924	55.67
Tangail	llama-3.1-8b-instant	3.82	27.94	0.6861	85.29
Tangail	llama-3.3-70b-versatile	25.98	53.90	0.7980	53.99
Tangail	openaigpt-oss-120b	24.10	52.38	0.8135	52.73
Tangail	**openaigpt-oss-20b**	**29.63**	**56.54**	**0.8276**	**47.27**

Table 4: Complete Pipeline 1 (Transcript-Based) results for all LLMs across all dialects. Best performance for each dialect-metric combination is shown in bold. LLM names are bolded when they achieve the most wins across all metrics for that dialect.

Dialect	Model	BLEU	ChrF	BERTScore F1	WER ↓
Chittagong	gemini-2.5-flash	20.75	49.70	0.7766	60.37
Chittagong	gemini-2.5-flash-lite	28.58	53.85	0.7886	52.90
Chittagong	gemma-3-12b-it	20.22	48.96	0.7551	59.96
Chittagong	**gemma-3-27b-it**	**57.14**	**72.52**	**0.8603**	**32.37**
Chittagong	gemma-3n-e4b-it	21.57	50.51	0.7415	58.09
Chittagong	llama-3.1-8b-instant	25.41	50.66	0.7334	56.22
Chittagong	llama-3.3-70b-versatile	25.93	53.12	0.7795	54.77
Chittagong	llama-4-scout-17b-16e-instruct	25.38	52.46	0.7581	54.98
Chittagong	openaigpt-oss-120b	17.00	47.09	0.7523	62.03
Chittagong	openaigpt-oss-20b	19.73	47.60	0.7158	59.34
Habiganj	gemini-2.5-flash	31.43	60.33	0.7850	43.22
Habiganj	gemini-2.5-flash-lite	27.52	57.57	0.7831	46.82
Habiganj	gemma-3-12b-it	25.90	54.66	0.7529	48.94
Habiganj	**gemma-3-27b-it**	**35.07**	**62.14**	**0.8023**	**41.31**
Habiganj	gemma-3n-e4b-it	22.05	53.78	0.7303	54.03
Habiganj	llama-3.1-8b-instant	23.01	52.10	0.7250	54.24
Habiganj	llama-3.3-70b-versatile	25.97	56.66	0.7673	48.52
Habiganj	llama-4-scout-17b-16e-instruct	27.97	58.96	0.7742	45.76
Habiganj	openaigpt-oss-120b	23.07	53.18	0.7249	50.64
Habiganj	openaigpt-oss-20b	23.64	53.36	0.7341	50.85
Rangpur	gemini-2.5-flash	22.28	53.68	**0.8224**	55.58
Rangpur	gemini-2.5-flash-lite	19.77	50.11	0.7871	57.94
Rangpur	gemma-3-12b-it	21.73	54.11	0.7959	53.43
Rangpur	**gemma-3-27b-it**	**28.38**	**57.24**	0.8144	**49.79**
Rangpur	gemma-3n-e4b-it	18.85	52.32	0.7809	58.15
Rangpur	llama-3.1-8b-instant	21.74	52.13	0.7813	57.30
Rangpur	llama-3.3-70b-versatile	19.98	52.75	0.8079	55.36
Rangpur	llama-4-scout-17b-16e-instruct	19.67	51.96	0.8169	57.30
Rangpur	openaigpt-oss-120b	14.97	48.61	0.7968	60.94
Rangpur	openaigpt-oss-20b	19.59	51.59	0.7938	57.73
Tangail	gemini-2.5-flash	48.00	72.99	0.8762	32.35
Tangail	gemini-2.5-flash-lite	51.32	72.65	0.8786	30.88
Tangail	gemma-3-12b-it	43.48	67.06	0.8447	35.92
Tangail	**gemma-3-27b-it**	**59.65**	**78.15**	**0.9143**	**23.32**
Tangail	gemma-3n-e4b-it	35.41	64.28	0.8562	43.07
Tangail	llama-3.1-8b-instant	40.16	66.19	0.8598	36.97
Tangail	llama-3.3-70b-versatile	45.35	68.43	0.8689	33.82
Tangail	llama-4-scout-17b-16e-instruct	38.57	67.04	0.8431	38.24
Tangail	openaigpt-oss-120b	40.69	67.02	0.8669	36.97
Tangail	openaigpt-oss-20b	37.45	65.19	0.8503	39.50

Table 5: Complete Pipeline 2 (Standardized Sentence-Pairs) results for all LLMs across all dialects. Best performance for each dialect-metric combination is shown in bold. LLM names are bolded when they achieve the most wins across all metrics for that dialect.

Dialect	Model	BLEU	ChrF	BERTScore F1	WER ↓
Chittagong	gemini-2.5-flash	4.68	28.29	0.6153	81.74
Chittagong	gemini-2.5-flash-lite	5.22	28.75	0.6271	78.84
Chittagong	gemma-3-12b-it	4.60	24.95	0.6033	85.68
Chittagong	gemma-3-27b-it	4.91	27.99	0.6177	79.46
Chittagong	gemma-3n-e4b-it	4.48	26.71	0.6200	80.08
Chittagong	llama-3.1-8b-instant	4.12	27.73	0.5889	83.40
Chittagong	llama-3.3-70b-versatile	4.75	30.66	0.6262	77.18
Chittagong	llama-4-scout-17b-16e-instruct	6.33	32.60	0.6113	76.56
Chittagong	openaigpt-oss-120b	6.44	30.95	**0.6500**	77.18
Chittagong	**openaigpt-oss-20b**	**7.72**	**36.19**	0.6338	**71.37**
Comilla	**gemini-2.5-flash**	**18.94**	**44.34**	**0.7336**	**60.63**
Comilla	gemini-2.5-flash-lite	13.85	37.74	0.7253	68.84
Comilla	gemma-3-12b-it	5.24	26.10	0.6238	84.63
Comilla	gemma-3-27b-it	4.83	28.49	0.6711	79.58
Comilla	gemma-3n-e4b-it	5.76	26.86	0.6704	80.42
Comilla	llama-3.1-8b-instant	3.44	27.99	0.6383	91.37
Comilla	llama-3.3-70b-versatile	7.97	33.59	0.6976	73.26
Comilla	llama-4-scout-17b-16e-instruct	8.09	31.24	0.6652	79.79
Comilla	openaigpt-oss-120b	7.04	34.49	0.7010	73.47
Comilla	openaigpt-oss-20b	5.53	31.55	0.6760	77.05
Habiganj	gemini-2.5-flash	5.54	34.58	0.6304	73.73
Habiganj	gemini-2.5-flash-lite	7.55	33.19	0.6263	74.58
Habiganj	gemma-3-12b-it	3.15	23.25	0.5664	86.44
Habiganj	gemma-3-27b-it	6.30	28.34	0.6035	78.39
Habiganj	gemma-3n-e4b-it	4.81	27.86	0.6300	78.39
Habiganj	llama-3.1-8b-instant	2.32	25.35	0.5755	83.90
Habiganj	llama-3.3-70b-versatile	6.67	35.41	0.6250	70.97
Habiganj	llama-4-scout-17b-16e-instruct	6.84	33.04	0.6034	75.00
Habiganj	openaigpt-oss-120b	7.89	32.64	0.6191	74.58
Habiganj	**openaigpt-oss-20b**	**8.72**	**37.04**	**0.6334**	**70.13**
Rangpur	gemini-2.5-flash	8.81	33.03	0.7193	75.97
Rangpur	gemini-2.5-flash-lite	**14.48**	38.77	0.7135	67.60
Rangpur	gemma-3-12b-it	4.14	27.78	0.6092	80.90
Rangpur	gemma-3-27b-it	7.65	30.79	0.6575	76.61
Rangpur	gemma-3n-e4b-it	6.15	35.47	0.7476	73.18
Rangpur	llama-3.1-8b-instant	5.25	34.32	0.6923	75.32
Rangpur	llama-3.3-70b-versatile	9.22	37.45	0.7173	69.74
Rangpur	llama-4-scout-17b-16e-instruct	6.44	32.37	0.6780	76.82
Rangpur	openaigpt-oss-120b	7.15	35.96	0.7453	72.32
Rangpur	**openaigpt-oss-20b**	14.30	**46.29**	**0.7656**	**62.02**
Sylhet	**gemini-2.5-flash**	**15.61**	**45.06**	**0.7077**	**62.00**
Sylhet	gemini-2.5-flash-lite	8.11	33.91	0.6960	74.53
Sylhet	gemma-3-12b-it	2.78	23.42	0.5770	88.94
Sylhet	gemma-3-27b-it	4.58	28.89	0.6274	80.79
Sylhet	gemma-3n-e4b-it	4.80	25.93	0.6480	81.00
Sylhet	llama-3.1-8b-instant	1.27	23.71	0.6092	87.68
Sylhet	llama-3.3-70b-versatile	4.29	34.86	0.6534	74.11
Sylhet	llama-4-scout-17b-16e-instruct	2.07	26.50	0.6073	85.39
Sylhet	openaigpt-oss-120b	3.20	31.12	0.6674	79.33
Sylhet	openaigpt-oss-20b	4.02	32.91	0.6682	77.66
Tangail	gemini-2.5-flash	18.80	47.47	0.7918	56.72
Tangail	gemini-2.5-flash-lite	19.90	46.50	0.7790	57.14
Tangail	gemma-3-12b-it	11.34	32.14	0.6553	74.79
Tangail	gemma-3-27b-it	13.47	38.01	0.7205	64.92
Tangail	gemma-3n-e4b-it	15.57	41.20	0.7822	62.82
Tangail	llama-3.1-8b-instant	15.70	48.95	0.7217	72.69
Tangail	llama-3.3-70b-versatile	20.59	47.78	0.7864	55.88
Tangail	llama-4-scout-17b-16e-instruct	13.31	40.21	0.7256	64.08
Tangail	openaigpt-oss-120b	22.05	49.29	**0.8183**	54.41
Tangail	**openaigpt-oss-20b**	**30.05**	**57.60**	0.8125	**46.01**

Table 6: Complete Zero-Shot results for all LLMs across all dialects. Best performance for each dialect is shown in bold.

Exploring Cross-Lingual Knowledge Transfer via Transliteration-Based MLM Fine-Tuning for Critically Low-resource Chakma Language

Adity Khisa
IIT, University of Dhaka
bsse1334@iit.du.ac.bd

Nusrat Jahan Lia
IIT, University of Dhaka
bsse1306@iit.du.ac.bd

Tasnim Mahfuz Nafis
IIT, University of Dhaka
bsse1327@iit.du.ac.bd

Zarif Masud
Toronto Metropolitan University
zarif.masud@gmail.com

Tanzir Pial
Stony Brook University
tpial@cs.stonybrook.edu

Shebuti Rayana
State University of New York at Old Westbury
rayanas@oldwestbury.edu

Ahmedul Kabir
IIT, University of Dhaka
kabir@iit.du.ac.bd

Abstract

As an Indo-Aryan language with limited available data, Chakma remains largely underrepresented in language models. In this work, we introduce a novel corpus of contextually coherent Bangla-transliterated Chakma, curated from Chakma literature, and validated by native speakers. Using this dataset, we fine-tune six encoder-based transformer models, including multilingual (mBERT, XLM-RoBERTa, DistilBERT), regional (BanglaBERT, IndicBERT), and monolingual English (DeBERTaV3) variants on masked language modeling (MLM) tasks. Our experiments show that fine-tuned multilingual models outperform their pre-trained counterparts when adapted to Bangla-transliterated Chakma, achieving up to 73.54% token accuracy and a perplexity as low as 2.90. Our analysis further highlights the impact of data quality on model performance and shows the limitations of OCR pipelines for morphologically rich Indic scripts. Our research demonstrates that Bangla-transliterated Chakma can be very effective for transfer learning for Chakma language, and we release our dataset[1] to encourage further research on multilingual language modeling for low-resource languages.

1 Introduction

Large Language Models (LLMs) have transformed the Natural Language Processing (NLP) world through unsupervised pre-training using large corpora of unlabeled data. Since labeled data are not required, LLMs can take advantage of the huge text corpora available in the public domain. For example, even first-generation language models such as BERT use a corpus of 3.3 billion English words (Devlin et al., 2019), while more recent LLMs use multiple massive corpora such as RedPajama (Weber et al., 2024) scraped from the web with hundreds of trillions of tokens. However, the sheer volume of data required for pre-training LLMs poses a challenge for low-resource languages even without labels, as seen in some recent works using datasets of 15 million words for Māori (James et al., 2022), 332 million tokens for Swahili (Conneau et al., 2020), and 108 million tokens from 11 African languages for AfriBERTa (Ogueji et al., 2021). Compared to the trillions of tokens available in high-resource languages, these million-scale corpora are minuscule. Consequently, training LLMs with low-resource corpora does not yield good results, as upon encountering new vocabulary, expressions, or culturally specific semantics, the models struggle to utilize their training patterns for accurate understanding and generation (Zhong et al., 2024).

To address this limitation, researchers have explored knowledge transfer, from LLMs trained on high-resource languages through Masked Language Model (MLM) fine-tuning on the comparatively lower resource language corpus (Fernando and Ranathunga, 2025). Muller et al. (2020) further showed that we can leverage the same transfer learning benefit through transliteration when the two languages do not share a script. They achieved a significant performance gain for Uyghur (105K sentences) and Sorani Kurdish (380K sentences) transliterated into the Latin script, compared to pre-training on those data in their original script alone.

[1] https://github.com/adity1234567/Chakma-MLM-Dataset.git

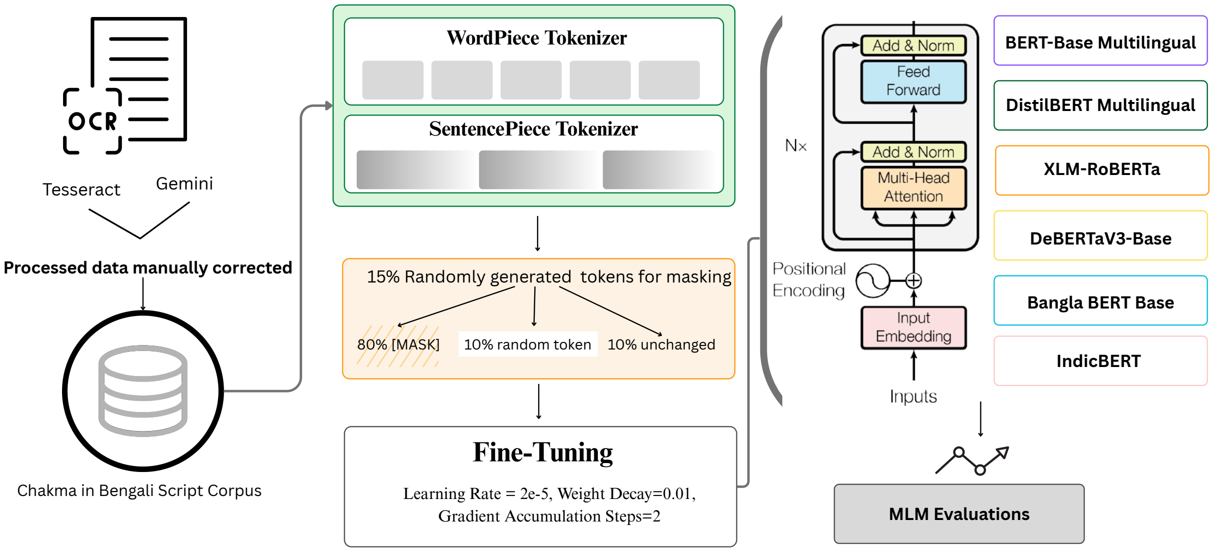


Figure 1: Overall workflow of OCR-based data curation, manual correction, and MLM fine-tuning for Bangla-transliterated Chakma language model

Chakma is an Indo-Aryan language, used as a first language by roughly one million people from the Chakma community living across parts of Bangladesh, India and Myanmar (Chakma Autonomous District Council, 2025). Although Chakma has its own script Ojhā Pāṭh, a considerable portion of Chakma literature is produced in Bangla transliteration (Brandt, 2018). Chakma remains a low-resource language with data scarcity both in its original script and Bangla transliteration (Chakma et al., 2024). At the same time, Bangla script is regularly used in training of multilingual LLMs like mBERT (Pires et al., 2019). In this context, our work shows that Bangla-transliterated Chakma dataset can yield moderately strong performance through MLM fine-tuning. Following Muller et al. (2020)'s idea for transliteration, we use Chakma text transliterated in Bangla for MLM-tuning multiple LLMs that are pre-trained on Bangla. Since, to the best of our knowledge, no contextually coherent Bangla-transliterated Chakma corpus exists, we have curated a novel corpus from Chakma books containing 4,570 manually validated sentences to run our experiments.

Our major contributions are as follows:

- We develop a novel Bangla-transliterated Chakma dataset, curated from images sourced from four books of Chakma literature using Tesseract OCR, comprising a total number of 6,353 sentences, of which 4,570 have been manually corrected.
- We show that language models can learn low-resource languages via MLM fine-tuning on the script of a related language, as demonstrated using Bangla script to fine-tune a Chakma model.
- We demonstrate how data quality impacts model performance, showing that better OCR for Bangla script, compatible with Bangla-transliterated Chakma, can significantly improve transfer learning for the Chakma language.

2 Related Works

In this section, we discuss four key areas that inform our work: multilingual NLP, model adaptation and quantization, tokenization and morphological challenges in Indic scripts, and existing language resources for Bangla and Chakma. These topics collectively highlight the progress and challenges in building effective models for low-resource languages like Chakma.

2.1 Multilingual NLP

The evolution of multilingual models has been driven by the need to extend transformer-based models to low-resource languages, particularly those with limited data or non-Latin scripts (Pakray et al., 2025). Devlin et al. (2019) introduced BERT along with its multilingual variant mBERT, and this marked a turning point. Models like mBERT, pre-trained on Wikipedia data across 104 languages using WordPiece (vocabulary of 110K tokens), and XLM-R, trained on CommonCrawl data from 100 languages with a 250K SentencePiece vocabulary (Conneau et al., 2019), enabled zero-shot cross-lingual transfer. XLM-R, relying solely on MLM pretraining, achieved state-of-the-art performance on multiple benchmarks (Ebrahimi and Kann, 2021).

These multilingual models often show strong zero-shot performance, but disparities remain: languages with less pretraining data or non-Latin scripts typically lag behind high-resource languages (Ebrahimi and Kann, 2021; Marchisio et al., 2024). For example, Wu and Dredze (2020) and Muller et al. (2020) show that mBERT's zero-shot accuracy varies widely by language, with some "hard" languages (often low-resource or using different scripts) remaining poorly served without additional adaptation. These findings spurred research leveraging pretrained transformer models and specialized techniques to handle underrepresented languages (Tela et al., 2020; Hangya et al., 2022; Bharadiya, 2023; Pakray et al., 2025). Our work builds on this by fine-tuning a Chakma-specific MLM encoder, addressing data scarcity for this low-resource Indic language.

2.2 Model adaptation techniques and quantization

To address performance disparities in low-resource languages, adaptation strategies emerged to tailor pre-trained models to specific languages or domains. When more data are available, continued monolingual pre-training in target-language data, as demonstrated by Chau et al. (2020), improved zero-shot performance, while domain-adaptive MLM pre-training improves downstream performance even in low-resource settings (Gururangan et al., 2020). Another strategy is to expand the vocabulary of a multilingual model to better cover the target language's lexicon and then additional MLM training improves performance for underrepresented languages (Wang et al., 2020).

2.3 Tokenization and morphology in Indic scripts

Indic languages like Bangla and Chakma are morphologically rich and use complex abugida scripts (Chowdhury, 2025), which raise challenges for subword tokenization. Standard BPE or WordPiece tokenizers can fragment important morphological units, hurting model performance (Pattnayak et al., 2025). Recent work demonstrates that SentencePiece (unigram) tokenization often preserves morphological information better than BPE for Indic languages. For instance, Pattnayak et al. (2025) found that, for zero-shot named entity recognition across several Indic languages, a SentencePiece-based vocabulary outperformed BPE, because it more cleanly segments root words and affixes. Others have noted that vowel forms in abugida scripts (*matras*) attach to consonants and can appear above, below, or beside the base character, which makes character-level segmentation non-trivial (Kashid and Bhattacharyya, 2024; Maung et al., 2025).

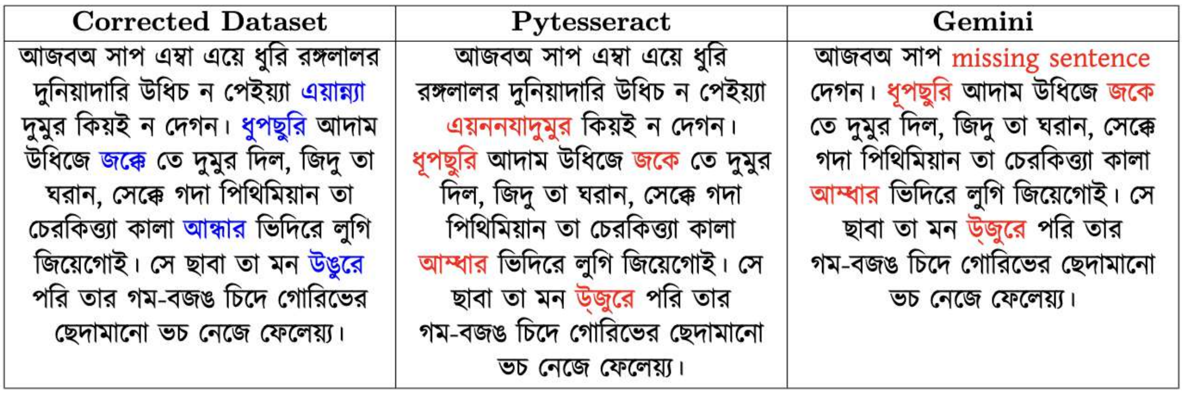
2.4 Bangla and Chakma language resources

Although Joshi et al. (2020) categorize Bangla among languages lacking labeled data, Bhattacharjee et al. (2021) developed BanglaBERT, a BERT-base model on the *Bangla2B+* corpus with 2.18 billion tokens from Bangla text, and introduced the Bangla Language Understanding Benchmark (BLUB). BanglaBERT achieves state-of-the-art results on multiple Bangla NLU tasks, outperforming both multilingual baselines (mBERT, XLM-R) and previous monolingual models.

In contrast, NLP work on Chakma is scant. The first known work in Chakma NLP effort is ChakmaNMT (Chakma et al., 2024), which constructed the first parallel corpus (15K sentence-pairs translation, from Chakma to Bangla) and trained a translation model. Using BanglaT5 and transliteration-based back-translation, they achieved a BLEU score of 17.8 for Chakma to Bangla translation. However, this work does not include the Bangla-transliterated Chakma text. The MELD dataset (Mahi et al., 2025) compiled transliterated sentence-level text in Chakma (and Garo, Marma) using the Bangla script. We opted not to use MELD, as its collection of isolated sentences lacks the semantic coherence required for our study. Instead, we focus on Bangla-transliterated Chakma texts extracted via OCR from printed literature.

3 Dataset Creation

Emphasizing the **authenticity** of linguistic resources, particularly in the field where the digitized materials are scarce and under-resourced, we con-



Corrected Dataset	Pytesseract	Gemini
আজবত্র সাপ এস্বা এয়ে ধুরি রঙ্গলালর দুনিয়াদারি উধিচ ন পেইয়্যা এয়ান্ন্যা দুমুর কিয়ই ন দেগন। ধুপছুরি আদাম উধিজে জক্কে তে দুমুর দিল, জিদু তা ঘরান, সেক্কে গদা পিথিমিয়ান তা চেরকিত্ত্যা কালা আন্ধার ভিদিরে লুগি জিয়েগোই। সে ছাবা তা মন উঙুরে পরি তার গম-বজঙ চিদে গোরিভের ছেদামানো ভচ নেজে ফেলেয়্য।	আজবত্র সাপ এস্বা এয়ে ধুরি রঙ্গলালর দুনিয়াদারি উধিচ ন পেইয়্যা এয়ননযাদুমুর কিয়ই ন দেগন। ধূপছুরি আদাম উধিজে জকে তে দুমুর দিল, জিদু তা ঘরান, সেক্কে গদা পিথিমিয়ান তা চেরকিত্ত্যা কালা আম্ধার ভিদিরে লুগি জিয়েগোই। সে ছাবা তা মন উ্জুরে পরি তার গম-বজঙ চিদে গোরিভের ছেদামানো ভচ নেজে ফেলেয়্য।	আজবত্র সাপ missing sentence দেগন। ধূপছুরি আদাম উধিজে জকে তে দুমুর দিল, জিদু তা ঘরান, সেক্কে গদা পিথিমিয়ান তা চেরকিত্ত্যা কালা আম্ধার ভিদিরে লুগি জিয়েগোই। সে ছাবা তা মন উ্জুরে পরি তার গম-বজঙ চিদে গোরিভের ছেদামানো ভচ নেজে ফেলেয়্য।

Figure 2: Sample data illustrating quality comparison across different methods, highlighting **missing sentences** in Gemini and **spelling errors** in other models caused by the misinterpretation of conjunct characters, phonetic signs, vowel diacritics, consonant modifiers, nasalization, and related orthographic features.

Model	Vocab Size	Tokenizer & Special Tokens	Tokenization Method
BERT-Base Multilingual (cased)	∼120k (WordPiece)	[CLS] ... [SEP], [MASK]	WordPiece
DistilBERT Multilingual (cased)	∼120k (WordPiece)	[CLS] ... [SEP], [MASK]	WordPiece
XLM-RoBERTa (XLM-R)	∼250k (SentencePiece)	$\langle s\rangle$ ... $\langle /s\rangle$, $\langle mask\rangle$	SentencePiece
DeBERTaV3-Base	∼128k (WordPiece-style)	[CLS] ... [SEP], [MASK]	WordPiece-style
Bangla BERT Base	∼32k (WordPiece)	[CLS] ... [SEP], [MASK]	WordPiece
IndicBERT	∼200k (SentencePiece)	$\langle s\rangle$ ... $\langle /s\rangle$, $\langle mask\rangle$	SentencePiece

Table 1: Comparison of encoder-based models used in our evaluation. Differences arise in vocabulary size, tokenizer conventions, and tokenization methods. The models include BERT-Base Multilingual (Devlin et al., 2019), DistilBERT Multilingual (Sanh et al., 2019), XLM-RoBERTa (Liu et al., 2019), DeBERTaV3-Base (He et al., 2021), Bangla BERT (Sarker, 2020), and IndicBERT (Kakwani et al., 2020).

struct a novel dataset combining four books (novels and poems) written in the Chakma language, utilizing Bangla script (see Figure 4 and Table 2). To collect these materials, we directly engaged with scholars whose first language is Chakma. The books were gathered from libraries on the basis of their recommendations. However, we acknowledge that most scholars prioritize the preservation and use of their own Chakma script. Chakma et al. (2024) also assigns importance to the Chakma script. Most pretrained models lack support for the complex structure of Chakma scripts. The Bangla-transliterated Chakma script enables the models to process the language effectively using their existing tokenizers.

3.1 Corpus Compilation: Sources and Scale

The data were extracted from the image of the pages of the books using **three primary methods**: Pytesseract (Hoffstaetter et al.), Gemini (Comanici et al., 2025), and manual processing. We used the PyTesseract OCR model and the Gemini 2.5 Pro model API separately to independently assess the quality of text extraction from different systems.

PyTesseract encountered problems with the recognition of Bangla's conjunctive characters and committed frequent spelling errors, as presented in Figure 2. On the other hand, Gemini 2.5 Pro with the free API, posed usage restrictions creating a barrier to scalability as we processed 400 images. Moreover, the Gemini API deviated from correctness in alphabet recognition, often produced incomplete sentences, and sometimes omitted entire sentences, affecting the overall quality of the extracted text. Some examples are presented in Figure 2.

Due to these limitations, we manually fixed one book entirely and another book partially, which we discuss in Section 3.2. The dataset is split into training, testing and validation subsets, as shown in Table 2.

3.2 Manual Curation for Linguistic Fidelity

Both the OCR models and LLMs struggled with accurate processing of conjunct characters, phonetic signs, including vowel diacritics, consonant modifiers, nasalization, silent consonant and incom-

Dataset	Dataset Split (sentences)
Tesseract OCR (Tes OCR)	train: 4,348 eval: 832 test: 1,173
Gemini OCR	train: 3,815 eval: 994 test: 1,173
Manually Fixed Data	train: 2,908 eval: 545 test: 1,118

Table 2: Breakdown of training, evaluation, and test sentence counts for datasets obtained from Tesseract OCR, Gemini OCR outputs, and manually corrected data.

pleteness of sentences (Figure 2). These complex character clusters are fundamental to the Bangla orthography but often are misinterpreted or omitted by OCR systems due to their non-linear composition and script variability (Ali et al., 2023; Guo et al., 2023). After identifying the limitations and to ensure the linguistic fidelity of our dataset, 4,570 sentences of OCR and LLM extracted text underwent a multi-stage manual correction and validation process. Two co-authors of this paper rectified these specific errors to guarantee the high integrity and usability of the final manual dataset. The overall workflow of the paper is presented in Figure 1.

4 MLM-tuning for low-resource languages

We fine-tuned six encoder-based models (including monolingual, multilingual and regional variants) on limited Chakma text written in Bangla script and compared their performance. Table 1 summarizes the models used in our experiments, including their vocabulary sizes and tokenization algorithms. All of these models have been pre-trained on Bangla before.

These LLMS are trained to predict the probability of a masked token/word given the context of surrounding words. This gives the models a foundational understanding of trained languages that can be generalized to other tasks (Wolf et al., 2020). Although each model comes in a similar-sized 12-layer configuration (270M–300M parameters for base models), they vary in vocabulary sizes, tokenizer types, special tokens, and critically, the dataset they were first pre-trained on.

For MLM fine-tuning, we masked 15% of tokens in each input sequence using the standard masking strategy: 80% replaced with the appropriate mask token ([MASK] or <mask>), 10% substituted with random vocabulary tokens, and 10% left unchanged. We maintained strict separation between training, validation, and test datasets across all experiments to prevent data leakage.

After multiple trials and errors, in our final configuration, we use the Adam optimizer, with a learning rate of 2×10^{-5} for all the models. The maximum number of epochs is 20. The dropout rate is 0.01. We keep the batch size at 8. For testing the fine-tuned models, we ensure consistency and reproducibility across the models.

5 Results

We evaluated the performance across three different data processing pipelines (Pytesseract, Gemini and manual processing) and also compared both universal and regional model types. Following the works of Salazar et al. (2019), Rogers et al. (2021) and Ethayarajh (2019), we evaluate our MLM fine-tuned models using perplexity, masked token accuracy, precision, recall, F1(macro), pseudo-log-likelihood (PLL) and predictive entropy.

5.1 Language Modeling Capability

RQ1: How effective are the pre-trained language models at masked language modeling for the (monolingual) Chakma language written in the Bangla script?

Table 3 shows that fine-tuned encoder-based language models consistently outperform their pre-trained counterparts for Bangla-transliterated Chakma. The fine-tuned models achieve accuracies up to 73.54% (XLM-RoBERTa) and perplexity as low as 2.899 (DeBERTaV3-Base) on manually corrected data, underscoring the value of adaptation for low-resource languages. Notably, monolingual models like DeBERTaV3-Base, which start with no prior knowledge of Chakma or Bangla script (0% baseline accuracy), achieve competitive results post-fine-tuning, demonstrating the robustness of adaptation even without cross-lingual pre-training. MLM-tuning also yields a marked reduction in prediction entropy, indicating increased confidence in masked-token predictions in Table 6.

Our best perplexity score of 2.899 is substantially lower (indicating better performance) than

Model	Accuracy (%) ↑			Perplexity ↓		
	Without MLM	With MLM	**Performance**	Without MLM	With MLM	**Performance**
DeBERTaV3-Base	0.00	72.08	+72.08	39329757.5	**2.90**	-39329754.6
XLM-RoBERTa	46.24	**73.54**	+27.30	24.39	3.27	-21.12
BERT-Base mBERT	**48.43**	70.00	+21.57	**13.12**	4.017	-9.103
DistilBERT Multilingual	38.78	65.08	+26.30	24.284	4.3046	-19.978
Bangla BERT Base	29.87	54.52	+24.65	250.09	11.79	-238.3
IndicBERT	17.54	45.36	+27.82	1823.61	16.79	-1806.82

Table 3: Performance comparison of models before and after MLM fine-tuning using manually annotated Chakma corpora. Accuracy (%) and perplexity are reported. Lower perplexity indicates better language modeling performance.

the perplexity scores reported for BERT on English datasets (Salazar et al., 2019). We treat this as an empirical observation rather than definitive evidence of superior absolute performance. This low perplexity may be an artifact of our dataset characteristics, including its relatively small size and the specific nature of the data (potentially featuring simpler or more repetitive linguistic structures compared to diverse English corpora). Hypotheses for this include reduced lexical diversity or script-specific tokenization efficiencies in Chakma, but the exact reasons remain unclear and could be explored in more detail in future work, perhaps by evaluating on larger, more varied Chakma datasets.

Outperforming of universal models over regional encoders: From Figure 3 and Table 4, we observe a consistent advantage for multilingual encoder models (XLM-RoBERTa, BERT-Base mBERT, DistilBERT Multilingual) and the monolingual DeBERTaV3-Base over regional encoder models (BanglaBERT, IndicBERT). Because tokenizers and vocabulary sizes differ across models, masked-language accuracy and perplexity are computed on model-specific tokenizations rather than an identical token sequence. This can potentially advantage models that produce fewer tokens per input, since they evaluate fewer positions and may face fewer rare-subword predictions. However, we argue that the comparison remains informative: the vocabularies are not extremely different, and the underlying dataset is identical for all models, and that accuracy is not simply determined by token count (see Table 1 and Table 3).

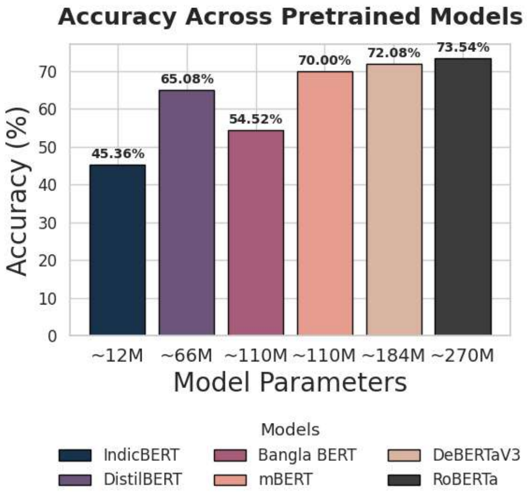


Figure 3: Comparison of universal multilingual and regional encoder models. Each grouped bar chart is showing the accuracy of pre-trained language models fine-tuned on manually fixed data, categorized by their parameter sizes.

We analyze two primary factors that influence model effectiveness: model parameter size and tokenization efficiency.

1. Parameter size → tokenization robustness. Larger multilingual models are trained on broader, more diverse corpora and typically learn richer subword vocabularies. This reduces out-of-vocabulary occurrences, over-fragmentation, and tokenization drift, which can otherwise harm downstream performance. These effects can cause some tokenizers to produce 2–3 times more tokens for the same input (see Figure 3) (Rust et al., 2020).

2. Tokenizer efficiency → evaluation metrics. A smaller number of tokens allows each token to carry more semantic context and reduces prediction

Model	Manual		Tesseract				Gemini			
			Self-Finetuned		Manual-Finetuned		Self-Finetuned		Manual-Finetuned	
	Acc.(↑)	PPL(↓)	Acc.(↑)	PPL(↓)	Acc.(↑)	PPL(↓)	Acc.(↑)	PPL(↓)	Acc.(↑)	PPL(↓)
DeBERTaV3-Base	72.08	2.90	46.94	7.82	46.52	9.75	46.86	9.12	45.77	10.01
XLM-RoBERTa	73.54	3.27	30.28	54.39	29.14	77.74	28.70	80.16	29.67	78.04
BERT-Base mBERT	70.00	4.02	32.91	27.50	31.56	40.59	31.75	44.16	31.23	42.35
DistilBERT Multilingual	65.08	4.30	31.64	29.05	29.63	43.91	30.49	43.20	30.04	41.96
Bangla BERT Base	54.52	11.79	22.17	299.54	20.19	384.28	20.71	483.57	20.86	467.25
IndicBERT	45.36	16.79	23.04	83.67	24.05	80.18	23.64	97.62	23.12	103.10

Table 4: Impact of data quality on model performance. Accuracy (%) and Perplexity (PPL) are reported for each model fine-tuned on manually annotated, Tesseract-processed, and Gemini-processed data. In our table, *Self-Finetuned* refers to training and evaluating each model on the same dataset, while *Manual-Finetuned* involves training on manually corrected data but evaluating on other test datasets like Tesseract or Gemini test sets.

noise for masked positions. Over-fragmentation, by contrast, spreads probability mass across many rare subwords, penalizing sequence-level scoring and hurting pseudo-log-likelihood (PLL) (Kudo and Richardson, 2018).

5.2 Impact of Data Quality

RQ2: In the context of the morphologically rich Bangla-transliterated Chakma, how does the OCR noise of data affect MLM performance?

Building on the findings from RQ1, where fine-tuning encoder-based models on manually corrected Chakma data demonstrated strong improvements in masked language modeling capabilities, we now explore the extent to which OCR-induced noise (stemming from script-specific challenges like transliteration variations and complex conjunct consonants) disrupts the learning of morphological structures in Bangla transliterated Chakma. In each case, the models were fine-tuned and evaluated on their respective dataset, which we refer to as *Self-Finetuned* in our Table 4. Additionally, we evaluated the model fine-tuned on the manually corrected dataset against the Tesseract and Gemini 2.5 Pro test sets, which we denote as *Manually-Finetuned* in the Table 4. Due to transliteration-induced variation with more complex conjunct consonants, the transliterated data (Bangla-transliterated Chakma) appears morphologically heavier than Bangla.

From the Table 4, we can see that models trained with Tesseract and Gemini 2.5 Pro processed data struggled to grasp the Chakma language, showing limited improvements even after fine-tuning, particularly evident in cases where models like DeBERTaV3-Base(He et al., 2021) had no initial understanding of Chakma (Table 3). The fine-tuning with the manually fixed dataset led to substantial gains in accuracy, highlighting that the model learns the affixes, inflections and complex forms of the language in a better way. Meanwhile, these models drop their performance when testing on the noisy test dataset. For instance, the XLM-RoBERTa model achieved its strongest performance with manual data, far surpassing its baseline and revealing that noisy OCR outputs can actually degrade model capabilities compared to their pre-fine-tuned state.

We find a similar pattern when examining perplexity across datasets for individual models. From Table 4, the manual dataset consistently yielded low perplexity, indicating strong language modeling and coherence. However, Tesseract and Gemini data introduced higher perplexity, often worsening it beyond the base model's levels due to inherent noise and errors. This trend holds across all six models in our experiments, emphasizing how high-quality data refines predictions while OCR-generated inaccuracies amplify confusion. Furthermore, when testing manually fine-tuned models on Tesseract or Gemini data, their perplexity suffered slightly compared to self-fine-tuned counterparts, reinforcing the pervasive impact of noise in OCR pipelines on overall model robustness.

Overall, these results show the critical role of preserving morphologically accurate data quality in enhancing model performance for low-resource indigenous languages like Chakma.

6 Conclusion

In this work, we introduced a Bangla-transliterated Chakma dataset, derived from Chakma literature using Tesseract, Gemini 2.5 Pro OCR and manual transcription. We empirically demonstrate that pre-trained multilingual language models can be effectively adapted for the Chakma language through fine-tuning on this data, establishing a strong baseline for Masked Language Modeling for Chakma. Our comprehensive experiments further underscore that model performance is highly sensitive to data quality, and that iterative cleaning directly enhances model performance. To support future research, we publicly release our manually refined dataset. A compelling direction for future work is to investigate the optimal transliteration target for low-resource languages. We hypothesize that for Chakma, which shares significant typological and lexical similarity with Bangla, transliteration into the Bangla script may yield superior performance compared to the English script, despite the generally stronger pre-training of LLMs on English. Systematically evaluating this trade-off between linguistic proximity and model capability remains an open question.

7 Limitations and Future Work

This study focuses on understanding the potential of LLM adaptability to low-resource languages. In our work, we have considered Chakma language as a case study. However, our manually validated Bangla-transliterated Chakma language dataset contains only 4570 sentences. The sentences are collected from story books, which is not sufficient to reflect diverse real-world scenarios, especially in a modern context. So, we aim to expand our Chakma corpus incorporating more diverse text sources, including spoken language transcripts, community-generated contents and parallel translations. Transliteration of Chakma dataset to Latin script is another direction of research following the works of Muller et al. (2020). If such a dataset exists, we can test the hypothesis that transliterating Chakma to a related language (Bangla) as opposed to the strongest language (English) may yield better performance. Inspired by Devlin et al. (2019), we can test our fine-tuned model for Next Sentence Prediction (NSP) accuracy to get a better understanding of how well our model is understanding the Chakma language. Improving OCR accuracy to extract the text with a better performance for conjunct characters, phonetic signs including vowel diacritics, consonant modifiers, nasalization, and others is also a potential direction for improvement.

8 Acknowledgement

The authors would like to thank Bijoy Chakma (Sociology, University of Rajshahi) and Shriswa Chakma (Department of Management Studies, Bangladesh University of Professionals) for their contributions to data collection and manual processing.

References

Hasmot Ali, AKM Shahariar Azad Rabby, Md Majedul Islam, A.k.m Mahamud, Nazmul Hasan, and Fuad Rahman. 2023. Gold standard Bangla OCR dataset: An in-depth look at data preprocessing and annotation processes. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 460–470, Singapore. Association for Computational Linguistics.

J Bharadiya. 2023. Transfer learning in natural language processing (nlp). *European Journal of Technology*, 7(2):26–35.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

Carmen Brandt. 2018. Writing off domination: the chakma and meitei script movements. *South Asian History and Culture*.

Aunabil Chakma, Aditya Chakma, Soham Khisa, Chumui Tripura, Masum Hasan, and Rifat Shahriyar. 2024. Chakmanmt: A low-resource machine translation on chakma language. *arXiv preprint arXiv:2410.10219*.

Chakma Autonomous District Council. 2025. The chakma people. https://www.cadc.gov.in/the-chakma-people/. Accessed: 2025-10-05.

Ethan C Chau, Lucy H Lin, and Noah A Smith. 2020. Parsing with multilingual bert, a small corpus, and a small treebank. *arXiv preprint arXiv:2009.14124*.

Nilanjana Roy Chowdhury. 2025. Phonological adaptations of bangla words in chakma. *Strength for Today and Bright Hope for Tomorrow Volume 25: 3 March 2025 ISSN 1930-2940*, page 58.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

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. *arXiv preprint arXiv:1911.02116*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Abteen Ebrahimi and Katharina Kann. 2021. How to adapt your pretrained multilingual model to 1600 languages. *arXiv preprint arXiv:2106.02124*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Aloka Fernando and Surangika Ranathunga. 2025. Linguistic entity masking to improve cross-lingual representation of multilingual language models for low-resource languages. *arXiv preprint arXiv:2501.05700*.

Linsheng Guo, Md Habibur Sifat, and Tashin Ahmed. 2023. Pipeline enabling zero-shot classification for Bangla handwritten grapheme. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 26–33, Singapore. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Viktor Hangya, Hossain Shaikh Saadi, and Alexander Fraser. 2022. Improving low-resource languages in pre-trained multilingual language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11993–12006.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Samuel Hoffstaetter, Juarez Bochi, Matthias Lee, Lars Kistner, Ryan Mitchell, and Emilio Cecchini. Python-tesseract: Ocr with tesseract in python. `https://github.com/h/pytesseract`,`https://tesseract-ocr.github.io/`.

Jesin James, Vithya Yogarajan, Isabella Shields, Catherine Watson, Peter J Keegan, Peter-Lucas Jones, and Keoni Mahelona. 2022. Language models for code-switch detection of te reo māori and english in a low-resource setting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 650–660. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. *arXiv preprint arXiv:2004.09095*.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul NC, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020. Indicnlpsuite: 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.

Harshvivek Kashid and Pushpak Bhattacharyya. 2024. Roundtripocr: A data generation technique for enhancing post-ocr error correction in low-resource devanagari languages. *arXiv preprint arXiv:2412.15248*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

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 preprint arXiv:1907.11692*.

Mehraj Hossain Mahi, Anzir Rahman Khan, Mobashsher Hasan Anik, Sheak Rashed Haider Noori, Arif Mahmud, and Mayen Uddin Mojumdar. 2025. Meld: A multilingual ethnic dataset of chakma, garo, and marma in bengali script with english and standard bengali translation. *Data in Brief*, page 111745.

Kelly Marchisio, Saurabh Dash, Hongyu Chen, Dennis Aumiller, Ahmet Üstün, Sara Hooker, and Sebastian Ruder. 2024. How does quantization affect multilingual llms? *arXiv preprint arXiv:2407.03211*.

Aye T Maung, Sumaiya Salekin, and Mohammad A Haque. 2025. A hybrid approach to bangla handwritten ocr: combining yolo and an advanced cnn. *Discover Artificial Intelligence*, 5(1):119.

Benjamin Muller, Antonis Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2020. When being unseen from mbert is just the beginning: Handling new languages with multilingual language models. *arXiv preprint arXiv:2010.12858*.

Kelechi Ogueji, Yuxin Zhu, and Jimmy Lin. 2021. Small data? no problem! exploring the viability of pretrained multilingual language models for low-resource languages. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 116–126. Association for Computational Linguistics.

Partha Pakray, Alexander Gelbukh, and Sivaji Bandyopadhyay. 2025. Natural language processing applications for low-resource languages. *Natural Language Processing*, 31(2):183–197.

Priyaranjan Pattnayak, Hitesh Patel, and Amit Agarwal. 2025. Tokenization matters: Improving zero-shot ner for indic languages. In *2025 IEEE International Conference on Electro Information Technology (eIT)*, pages 456–462. IEEE.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the association for computational linguistics*, 8:842–866.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2020. How good is your tokenizer? on the monolingual performance of multilingual language models. *arXiv preprint arXiv:2012.15613*.

Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. 2019. Masked language model scoring. *arXiv preprint arXiv:1910.14659*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sagor Sarker. 2020. Banglabert: Bengali mask language model for bengali language understanding.

Abrhalei Tela, Abraham Woubie, and Ville Hautamaki. 2020. Transferring monolingual model to low-resource language: The case of tigrinya. *arXiv preprint arXiv:2006.07698*.

Zihan Wang, Stephen Mayhew, Dan Roth, and 1 others. 2020. Extending multilingual bert to low-resource languages. *arXiv preprint arXiv:2004.13640*.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: A reproduction of the llama training dataset. *arXiv preprint arXiv:2411.12372*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual bert? *arXiv preprint arXiv:2005.09093*.

Tianyang Zhong, Zhenyuan Yang, Zhengliang Liu, Ruidong Zhang, Yiheng Liu, Haiyang Sun, Yi Pan, Yiwei Li, Yifan Zhou, Hanqi Jiang, and 1 others. 2024. Opportunities and challenges of large language models for low-resource languages in humanities research. *arXiv preprint arXiv:2412.04497*.

A Appendix

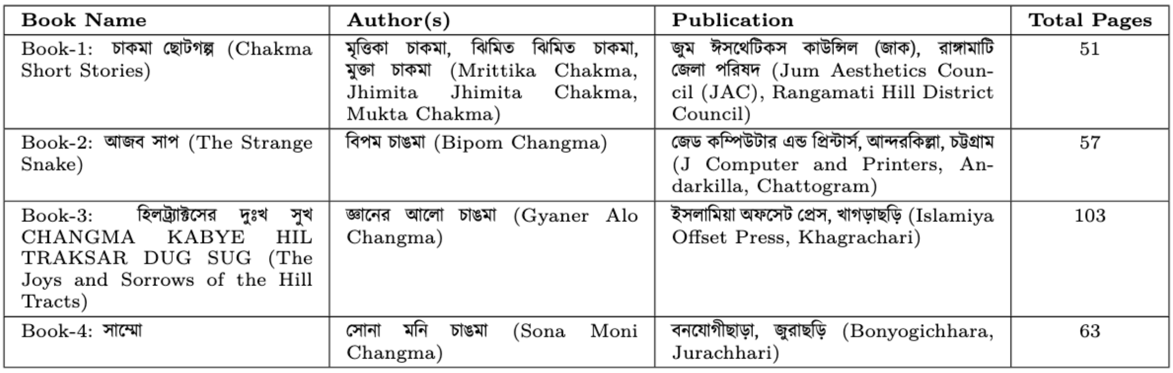
Table 5: Model Fine-tuned on Manual Data: Cross-Test Performance

Model	Data	Loss	Perplexity	Accuracy(%) (masked token acc.)	Precision	Recall	F1_macro	Prediction Entropy	Pseudo-log likelihood	Evaluated Tokens (sentences)
manual_roberta	manual	3.1942	24.3903	46.24	0.3432	0.2799	0.2876	2.8258	-3.1910	8092 (431)
	teserrect	4.3534	77.7443	29.14	0.2502	0.1469	0.1603	2.7436	-4.3550	10713 (569)
	gemini-2.5-pro	4.3573	78.0479	29.67	0.253	0.1576	0.1677	2.7591	-4.3559	10645 (569)
manual_bert	manual	1.3905	4.017	70	0.5107	0.4353	0.4512	1.0178	-1.3916	9011 (495)
	teserrect	3.7036	40.5924	31.56	0.2886	0.1837	0.1974	2.123	-3.7023	12198 (657)
	gemini-2.5-pro	3.7461	42.3548	31.23	0.2738	0.1716	0.1880	2.1382	-3.7426	12445 (657)
manual_distilbert	manual	1.4740	4.3668	67.10	0.4803	0.4074	0.4074	1.2122	-1.4742	9180 (495)
	teserrect	3.7822	43.9145	29.63	0.233	0.1415	0.1521	2.3025	-3.7787	12516 (657)
	gemini-2.5-pro	3.7368	41.9629	30.04	0.2357	0.1568	0.1636	2.3025	-3.7411	12424 (657)
manual_debarta	manual	1.0644	2.8991	72.08	0.452	0.3549	0.3724	0.8393	-1.0654	16202 (864)
	teserrect	2.2775	9.7526	46.52	0.1911	0.1402	0.1446	1.3831	-2.2782	20466 (1085)
	gemini-2.5-pro	2.3037	10.0116	45.77	0.1999	0.1460	0.1515	1.3957	-2.3066	20402 (1085)
sagorbangla	manual	2.4675	11.7925	54.52	0.367	0.2663	0.2861	2.6335	-2.4707	18587 (986)
	teserrect	6.1659	476.2059	20.45	0.1542	0.0979	0.1042	3.9245	-6.1638	8216 (434)
	gemini-2.5-pro	6.2530	519.5888	20.51	0.1343	0.0826	0.0886	3.9889	-6.2558	8148 (434)
IndicBERT	manual	2.4492	11.5791	52.75	0.4337	0.2994	0.3297	2.4188	-2.4546	6627 (351)
	teserrect	4.6562	105.2307	23.03	0.2572	0.1140	0.1385	3.7807	-4.6625	8392 (429)
	gemini-2.5-pro	4.5815	97.6632	23.19	0.2964	0.1357	0.1649	3.7785	-4.5871	8251 (429)

Table 6: Model Performance Comparison

Model	Data	Type	Loss	Perplexity	Accuracy (masked token accuracy)	Precision	Recall	F1_macro	Prediction Entropy	Pseudo-log-likelihood	Tokens
BERT-Base Multilingual (cased)	manual	base	2.5744	13.1234	48.43%	0.4173	0.3144	0.3321	2.3547	-2.5763	9126 (495)
		finetuned	1.3905	4.017	70%	0.5107	0.4353	0.4512	1.0178	-1.3916	9011 (495)
	teserrect	base	3.8187	45.5459	25.05%	0.2491	0.1507	0.1640	3.3034	-3.8194	12371 (657)
		finetuned	3.3143	27.5039	32.91%	0.2882	0.1706	0.1889	2.6401	-3.3211	12493 (657)
	gemini-2.5-pro	base	3.7713	43.4369	26.19%	0.2492	0.1543	0.1695	3.2742	-3.7731	12338 (657)
		finetuned	3.7878	44.1608	31.75%	0.2391	0.1821	0.1855	2.1291	-3.7898	12353 (657)
DistilBERT Multilingual (cased)	manual	base	3.1898	24.284	38.78%	0.3030	0.2044	0.2135	3.2877	-3.1989	9106 (495)
		finetuned	1.4597	4.3046	65.08%	0.5070	0.4045	0.4323	1.4180	-1.4636	8796 (439)
	teserrect	base	4.2519	70.2371	20.10%	0.2064	0.1030	0.1090	3.8775	-4.2527	12126 (657)
		finetuned	3.3691	29.052	31.64%	0.2845	0.1699	0.1872	2.6530	-3.3700	1197 (657)
	gemini-2.5-pro	base	4.2025	66.8516	20.30%	0.2175	0.1087	0.1150	3.8539	-4.2038	12459 (657)
		finetuned	3.7658	43.197	30.49%	0.2533	0.1701	0.1811	2.3147	-3.7675	12353 (657)
XLM-RoBERTa (XLM-R)	manual	base	3.1942	24.3903	46.24%	0.3432	0.2799	0.2876	2.8258	-3.1910	8092 (431)
		finetuned	1.1858	3.2732	73.54%	0.5677	0.5114	0.5219	1.0693	-1.1918	8043 (431)
	teserrect	base	4.5293	92.6939	25.79%	0.2294	0.1498	0.1619	4.0378	-4.5321	10653 (569)
		finetuned	3.14	54.39	30.28%	0.2634	0.1554	0.1719	3.2284	-3.9981	10327 (569)
	gemini-2.5-pro	base	4.5354	93.2627	25.72%	0.2213	0.1417	0.1552	4.0543	-4.5400	10762 (569)
		finetuned	4.3840	80.1618	28.70%	0.2421	0.1575	0.1631	2.8692	-4.3910	10620 (569)
DeBERTaV3 Base	manual	base	17.4875	39329757.5	0%	0.0000	0.0000	0.0000	7.8197	-17.4848	16457 (864)
		finetuned	1.0644	2.8991	72.08%	0.4520	0.3549	0.3724	0.8393	-1.0654	16202 (864)
	teserrect	base	14.6398	2280263.447	0%	0.0000	0.0000	0.0000	7.8999	-14.6415	20447 (1085)
		finetuned	2.0572	7.8241	46.94%	0.3072	0.1697	0.1859	1.562	-2.0610	20477 (1085)
	gemini-2.5-pro	base	16.1899	10744904.22	0%	0.0000	0.0000	0.0000	8.1224	-16.1899	20398 (1085)
		finetuned	2.2105	9.1200	46.86%	0.2412	0.1576	0.1688	1.4322	-2.2130	20509 (1085)
Bangla BERT Base	manual	base	5.5218	250.0904	29.87%	0.2338	0.1478	0.1637	5.7929	-5.5302	6451 (344)
		finetuned	2.4675	11.7925	54.52%	0.3670	0.2663	0.2861	2.6335	-2.4707	18587 (986)
	teserrect	base	6.8747	967.5296	15.67%	0.1516	0.0860	0.0981	6.6578	-6.8720	8129 (434)
		finetuned	5.7022	299.5393	22.17%	0.1598	0.1039	0.1116	4.2600	-5.7041	8167 (434)
	gemini-2.5-pro	base	6.8133	909.9072	16.95%	0.1514	0.0847	0.0967	6.5669	-6.8131	8256 (434)
		finetuned	6.1469	467.2527	20.86%	0.1442	0.0888	0.0953	3.9319	-6.1402	8179 (434)
IndicBERT	manual	base	7.5091	1824.6141	17.54%	0.3636	0.1131	0.1489	5.3301	-7.5190	18403 (971)
		finetuned	2.8209	16.7924	45.36%	0.4261	0.2239	0.2693	2.8646	-2.8273	18418 (971)
	teserrect	base	8.0048	2995.1897	12.62%	0.2703	0.0822	0.1060	5.4509	-8.0154	8065 (429)
		finetuned	4.4269	83.6677	23.04%	0.2744	0.1141	0.1434	3.9951	-4.4268	8046 (429)
	gemini-2.5-pro	base	8.0774	3220.7776	12.42%	0.2762	0.0784	0.1026	5.4241	-8.0825	8130 (429)
		finetuned	4.5811	97.6174	23.64%	0.2733	0.1186	0.1470	3.7354	-4.5813	8046 (429)

Figure 4: Details of Chakma Storybooks Used in the Dataset



Book Name	Author(s)	Publication	Total Pages
Book-1: চাকমা ছোটগল্প (Chakma Short Stories)	মৃত্তিকা চাকমা, ঝিমিত ঝিমিত চাকমা, মুক্তা চাকমা (Mrittika Chakma, Jhimita Jhimita Chakma, Mukta Chakma)	জুম ঈসথেটিকস কাউন্সিল (জাক), রাঙ্গামাটি জেলা পরিষদ (Jum Aesthetics Council (JAC), Rangamati Hill District Council)	51
Book-2: আজব সাপ (The Strange Snake)	বিপম চাঙমা (Bipom Changma)	জেড কম্পিউটার এন্ড প্রিন্টার্স, আন্দরকিল্লা, চট্টগ্রাম (J Computer and Printers, Andarkilla, Chattogram)	57
Book-3: হিলট্র্যাক্টসের দুঃখ সুখ CHANGMA KABYE HIL TRAKSAR DUG SUG (The Joys and Sorrows of the Hill Tracts)	জ্ঞানের আলো চাঙমা (Gyaner Alo Changma)	ইসলামিয়া অফসেট প্রেস, খাগড়াছড়ি (Islamiya Offset Press, Khagrachari)	103
Book-4: সাম্মো	সোনা মনি চাঙমা (Sona Moni Changma)	বনযোগীছাড়া, জুরাছড়ি (Bonyogichhara, Jurachhari)	63

LLMs for Low-Resource Dialect Translation Using Context-Aware Prompting: A Case Study on Sylheti

Tabia Tanzin Prama[1,2,3,4]
[1]Computational Story Lab, [2]Vermont Complex Systems Institute,
[3]Vermont Advanced Computing Center,
[4]Department of Computer Science,
University of Vermont, Burlington, VT 05405, USA

Abstract

Large Language Models (LLMs) have demonstrated strong translation abilities through prompting, even without task-specific training. However, their effectiveness in dialectal and low-resource contexts remains underexplored. This study presents the first systematic investigation of LLM-based Machine Translation (MT) for Sylheti, a dialect of Bangla that is itself low-resource. We evaluate five advanced LLMs (GPT-4.1, GPT-4.1, LLaMA 4, Grok 3, and Deepseek V3.2) across both translation directions (Bangla ⇔ Sylheti), and find that these models struggle with dialect-specific vocabulary. To address this, we introduce Sylheti-CAP (Context-Aware Prompting), a three-step framework that embeds a linguistic rulebook, dictionary (2260 core vocabulary and idioms), and authenticity check directly into prompts. Extensive experiments show that Sylheti-CAP consistently improves translation quality across models and prompting strategies. Both automatic metrics and human evaluations confirm its effectiveness, while qualitative analysis reveals notable reductions in hallucinations, ambiguities, and awkward phrasing—establishing Sylheti-CAP as a scalable solution for dialectal and low-resource MT. Dataset link: https://github.com/Sylheti-CAP

1 Introduction

Large Language Models (LLMs) have recently demonstrated remarkable potential in natural language processing (NLP) tasks (Yang et al., 2024; Dubey et al., 2024; OpenAI et al., 2023), including neural machine translation (NMT). Prior studies (Robinson et al., 2023; Zhu et al., 2023) show that while LLMs achieve strong performance in translating high-resource languages, their effectiveness decreases significantly for low-resource languages (LRLs) (Joulin et al., 2016; team et al., 2022), where parallel data is limited and difficult to obtain.

Compared to traditional NMT models, LLMs offer several qualitative advantages. They allow controllability of style and language variety through prompting and in-context learning (Brown et al., 2020; García et al., 2023; Agrawal et al., 2022), exhibit inherent document-level translation capabilities (Wang et al., 2023; Karpinska and Iyyer, 2023), produce less literal translations (Raunak et al., 2023), and demonstrate improved handling of complex linguistic phenomena such as idioms and ambiguous expressions. Consequently, LLMs are increasingly surpassing conventional NMT models in versatility (Peng et al., 2023; Hendy et al., 2023; Zhu et al., 2023).

Recent research has leveraged in-context learning (ICL) (Brown et al., 2020; Dong et al., 2022) to enable LLMs to perform translation without parameter updates, and supervised fine-tuning with parallel corpora has also been explored (Li et al., 2023; Chen et al., 2021; Alves et al., 2023). However, training LLMs still requires vast multilingual resources, and the inherent imbalance in language coverage continues to hinder performance for many LRLs (Jiao et al., 2023; Hendy et al., 2023). While prior work has shown impressive results in high-resource pairs such as English–German translation (Vilar et al., 2022), the effectiveness of LLMs in dialect-specific scenarios remains underexplored.

This gap is particularly acute for languages like Bangla (Prama et al., 2025). More than two hundred million people speak Bangla (also known as Bengali) (Accredited Language Services, 2015), yet it remains relatively low-resource in the NLP landscape. Its dialects are even more underserved, with virtually no large-scale datasets. These dialects encode rich linguistic and cultural variation, but unlike the standardized language, they rarely benefit from curated resources such as newswire corpora. Sylheti is a major Bangla dialect with an estimated 11 million speakers worldwide (Simard

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 292–308
December 23, 2025 ©2025 Association for Computational Linguistics

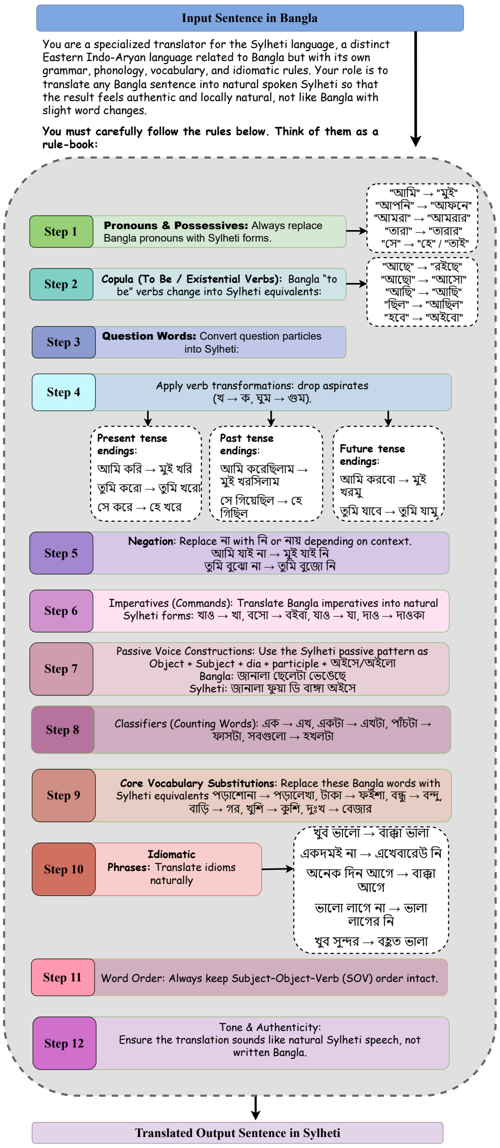
et al., 2020), illustrates this problem especially clearly. Although a few studies have explored Bangla⇔Sylheti translation using traditional deep learning models (Prama and Anwar, 2025a; Faria et al., 2023), research remains limited. To our knowledge, this is the first systematic evaluation of LLM-based machine translation for Bangla ⇔ Sylheti. We frame our study around two research questions (RQs):

RQ1: How do LLMs perform MT between Bangla and the Sylheti dialect?

To answer this question, we evaluate multilingual LLMs (LLaMA-4 (AI, 2024) , Gemini 2.5 Flash (DeepMind, 2025), GPT-4.1 (OpenAI, 2024), DeepSeek v3.2 (DeepSeek-AI, 2024), and Grok 3 (xAI, 2025)) from five different LLM families. Here the LLMs are first used in a zero-shot setting, meaning that we assume that (to the best of our knowledge) the models are not directly trained with Sylheti-specific data but are instead expected to apply their knowledge of Bangla to understand and translate Sylheti. On average, Sylheti → Bangla translation achieves 66.8 % higher BLEU-1 scores than Bangla→ Sylheti. Also Llama 4 and Grok achive superior perfromance among the models we tested.

RQ2: How can we improve LLM translation performance?

To address this question, we propose Sylheti-CAP, a context-aware prompting strategy designed to enhance LLM translation for low-resource dialects shows in Figure 1. While prior work has explored adding extra-sentential context to translation (Maruf et al., 2019; Castilho and Knowles, 2024), such models—trained solely for translation—have shown only modest gains over context-agnostic baselines (Chatterjee et al., 2020; Yin et al., 2021). Recent studies show that LLMs can effectively leverage contextual information for various NLP tasks, including document-level translation (Karpinska and Iyyer, 2023; Wang et al., 2023). Building on this, Sylheti-CAP integrates Sylheti-specific lexical, grammatical, and idiomatic knowledge (including untranslatable terms) directly into prompts, followed by a fluency and correctness refinement step. We evaluate Sylheti-CAP on Bangla ⇔ Sylheti translation using five LLMs. Results on BLEU, METEOR, and ChrF show consistent improvements over Zero-Shot, Few-Shot, and CoT prompting, with fewer mistranslations, omissions, and awkward phrases. Human preference and MQM (Lommel, 2013) evaluations further confirm that Sylheti-CAP yields more natural and faithful translations.



You are a specialized translator for the Sylheti language, a distinct Eastern Indo-Aryan language related to Bangla but with its own grammar, phonology, vocabulary, and idiomatic rules. Your role is to translate any Bangla sentence into natural spoken Sylheti so that the result feels authentic and locally natural, not like Bangla with slight word changes.

You must carefully follow the rules below. Think of them as a rule-book:

Step 1 **Pronouns & Possessives:** Always replace Bangla pronouns with Sylheti forms.

"আমি" → "মুই"
"আপনি" → "আফনে"
"আমরা" → "আমরার"
"তারা" → "তারার"
"সে" → "হে" / "তাই"

Step 2 **Copula (To Be / Existential Verbs):** Bangla "to be" verbs change into Sylheti equivalents:

"আছে" → "রইছে"
"আছো" → "আসো"
"আছি" → "আছি"
"ছিল" → "আছিল"
"হবে" → "অইবো"

Step 3 **Question Words:** Convert question particles into Sylheti:

Step 4 Apply verb transformations: drop aspirates (খ → ক, ঘুম → গুম).

Present tense endings:
আমি করি → মুই খরি
তুমি করো → তুমি খরো
সে করে → হে খরে

Past tense endings:
আমি করেছিলাম → মুই খরসিলাম
সে গিয়েছিল → হে গিছিল

Future tense endings:
আমি করবো → মুই খরমু
তুমি যাবে → তুমি যামু

Step 5 **Negation**: Replace না with নি or নায় depending on context.
আমি যাই না → মুই যাই নি
তুমি বুঝো না → তুমি বুজো নি

Step 6 Imperatives (Commands): Translate Bangla imperatives into natural Sylheti forms: খাও → খা, বসো → বইবা, যাও → যা, দাও → দাওকা

Step 7 Passive Voice Constructions: Use the Sylheti passive pattern as Object + Subject + dia + participle + অইসে/অইলো
Bangla: জানালা ছেলেটা ভেঙেছে
Sylheti: জানালা ফুয়া ডি বাঙ্গা অইসে

Step 8 Classifiers (Counting Words): এক → এখ, একটা → এখটা, পাঁচটা → ফাসটা, সবগুলো → হখলটা

Step 9 **Core Vocabulary Substitutions**: Replace these Bangla words with Sylheti equivalents পড়াশোনা → পড়ালেখা, টাকা → ফইশা, বন্ধু → বন্দু, বাড়ি → গর, খুশি → কুশি, দুঃখ → বেজার

Step 10 **Idiomatic Phrases:** Translate idioms naturally

খুব ভালো → বাক্কা ভালা
একদমই না → এখেবারেউ নি
অনেক দিন আগে → বাক্কা আগে
ভালো লাগে না → ভালা লাগের নি
খুব সুন্দর → বহুত ভালা

Step 11 Word Order: Always keep Subject-Object-Verb (SOV) order intact.

Step 12 Tone & Authenticity: Ensure the translation sounds like natural Sylheti speech, not written Bangla.

Figure 1: Overview of the Sylheti-CAP prompting framework. The framework consists of three key stages: (1) Linguistic Rulebook Integration with Sylheti-specific grammatical and morphological rules (2) Bilingual Lexicon and Idiom dictionary and (3) Authenticity and Fluency Check.

2 The Sylheti-CAP Framework: Prompting for Low-Resource Dialectal Translation

Prompting language models (LMs) for translation, particularly between standard and dialectal variants, assumes that the model has been pre-trained on sufficient parallel data in both languages. For low-resource languages like Sylheti, a dialect of Eastern Indo-Aryan Bangla with distinct phonology, grammar, and vocabulary, this assumption often fails—even in large multilingual LMs. Moreover, translation quality typically declines when faced with out-of-domain data (Zhang and Zong, 2016; Koehn and Knowles, 2017). To address these challenges of data scarcity and domain mismatch, we introduce the Sylheti-CAP (Sylheti Context-Aware Prompting) framework. This method leverages the in-context learning ability of LMs by injecting structured linguistic rules and bilingual lexicons directly into the translation prompt (Figure 1).

Dictionaries and rulebooks are often available even for low-resource languages, making them cost-effective sources of translation knowledge (Arthur et al., 2016; Zhong and Chiang, 2020; Hämäläinen and Alnajjar, 2019). The Sylheti-CAP framework integrates this information into the prompt through a three-part schema to ensure that outputs reflect authentic Sylheti usage rather than slightly modified Bangla.

Step 1. Linguistic Rulebook. This section defines the translator persona and the grammatical and phonological rules required for authentic Sylheti output. Key rules include:

- **Pronoun and Possessive Substitution:** আমি (I) → মুই (I), আমার (my) → মোর (my).
- **Copula and Existential Verbs:** আছে (is/are, exists) → রইছে (is/are, exists), আছো (you are) → আসো (you are).
- **Verb Transformations:** Apply phonological simplification (e.g., খ (kh sound) → ক (k sound)) and tense-specific conjugations (আমি করবো (I will do) → মুই খরমু (I will do)).
- **Syntactic and Morphological Directives:** Enforce negation (না (not) → নি/নায় (not)), imperatives (খাও (eat!) → খা (eat!)), and maintain SOV (Subject–Object–Verb) word order.

Step 2. Core Vocabulary and Idioms Dictionary. This section provides a lexicon of frequently used words and idiomatic expressions where direct translation is insufficient. It guides the model toward contextually appropriate substitutions and handles non-standard lexical gaps. A dictionary of 3,106 word pairs was created for this purpose (see Appendix A.1).

Core Vocabulary Examples:

Bangla Word	Sylheti
পড়াশোনা (study)	পড়ালেখা
টাকা (money)	ফইশা
বাড়ি (house/home)	গর
খুশি (happy)	কুশি
বন্ধু (friend)	বন্দু

Idiomatic Expressions:

Bangla Expression	Sylheti
খুব ভালো (very good)	বাক্কা ভালা
একদমই না (not at all)	এখেবারেউ নি
অনেক দিন আগে (a long time ago)	বাক্কা আগে
ভালো লাগে না (do not like / does not feel good)	ভালা লাগের নি

Step 3. Sentence-Level Translation and Authenticity Check. The final segment presents the Bangla source sentence, followed by meta-instructions guiding the model to prioritize fluency and natural spoken style over literal translation. This ensures the generated text reflects authentic Sylheti speech rather than formalized Bangla.

Overall, Sylheti-CAP combines linguistic rules and bilingual dictionaries within a structured prompt, providing an interpretable and adaptable method for high-quality dialect-specific machine translation—especially valuable for under-resourced language pairs where traditional neural MT systems fail to capture dialectal nuances. Appendix A.2 Table 12 shows the prompt we used following Sylheti-CAP framework.

3 Experiments

Dataset. For evaluation, we use the Vashantor corpus (Faria et al., 2023), which contains 2,500 Sylheti ⇔ Bangla parallel sentences collected from websites, social media platforms, and discussion boards. Each sentence has been professionally translated into Bangla. We use a 375-sentence test set to evaluate each model.

Dictionaries. For translation, we employ ground-truth bilingual dictionaries constructed from three Sylheti⇔Bangla parallel datasets: Vashantor (Faria et al., 2023) (2,125 sentences), ONUBAD (Sultana et al., 2025) (980 sentences), and a Sylheti dataset (Prama and Oni, 2025) (5,002 sentence pairs). From these sources, we derived word-level mappings by taking the union of unique tokens, resulting in 2260 distinct words that differ between the Sylheti and Bangla sides. Examples of Sylheti⇔Bangla word mappings are provided in Appendix A.1 (Table 7). Additionally, a large number of words are identical in both languages since Sylheti is a dialect of Bangla (see Table 8).

Models. We evaluated five state-of-the-art LLMs from major developers, each with distinct technical specifications. The selection prioritized cutting-edge, diverse architectures to enable a comprehensive competitive assessment.

LLaMA-4 (AI, 2024) comes in two variants: Llama 4 Scout (17B active parameters, 16 experts) and Llama 4 Maverick (17B active parameters, 128 experts). Llama 4 Maverick is considered the leading multimodal model in its class, outperforming GPT-4o, Gemini 2.0 Flash, and DeepSeek V3 on reasoning and coding benchmarks. In our experiments, we evaluate the Llama 4 Maverick model via the Meta.AI [1] website.

Gemini 2.5 Flash (DeepMind, 2025), (released on June 17, 2025) is Google's latest sparse mixture-of-experts Transformer model, optimized for large-context processing with up to 1,048,576 input tokens and 65,535 output tokens. It features advanced reasoning, agentic behaviors, and real-time application support. In this experiment, we evaluated Gemini 2.5 Flash using the Google AI Studio [2] platform.

GPT-4.1 (OpenAI, 2024) is a multimodal LLM that achieves human-level performance on diverse professional and academic benchmarks. Based on the Transformer architecture, it is pre-trained for next-token prediction and can process up to 32,768 tokens per input. The model is accessible via ChatGPT Plus and the OpenAI API; in this experiment, we accessed and evaluated GPT-4.1 through the OpenAI API.

Grok 3 (released February 17, 2025) (xAI, 2025) is xAI's latest 1.2-trillion-parameter model, combining transformer-based language modeling with symbolic reasoning modules (Inaba et al., 2003). It uses 128 expert networks with dynamic routing and cross-expert attention gates, achieving 83% parameter activation efficiency while enabling knowledge sharing between experts (Doshi et al., 2023) which is trained on 13.4 trillion tokens. In this experiment, we evaluated Grok 3 through its official web interface [3].

DeepSeek-V3 (released December 26, 2024) is a Mixture-of-Experts language model with 671 billion total parameters, 37 billion of which are active per token. It employs Multi-head Latent Attention (MLA) and the DeepSeekMoE architecture, extending DeepSeek-V2 for more efficient inference and cost-effective training. Pre-trained on 14.8 trillion tokens and further optimized via supervised finetuning and reinforcement learning. In this experiment, we evaluated DeepSeek-V3 using the official website [4].

Metrics. We evaluate LLM performance using BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002) and ChrF (Character-level F-score) (Popovic, 2015), which together offer a complementary view across tokenization granularities. In addition, we report METEOR (Banerjee and Lavie, 2005), which mitigates some semantic-matching limitations of BLEU by incorporating stemming and synonymy. Taken together, BLEU, ChrF, and METEOR provide a multi-dimensional assessment of translation quality.

Comparative Methods. We consider the following prompting strategies:

Zero-shot. A direct translation prompt with the model's default settings; temperature is set to 1 in all experiments.

Few-shot. In-context learning with exemplars included in the prompt (Hendy et al., 2023). Prior work shows that example selection strategy and count can affect performance (Agrawal et al., 2022; Zhu et al., 2023), with random selection often performing best (Zhu et al., 2023). As the number of examples increases from 1 to 8, BLEU typically improves (Zhu et al., 2023). We use five exemplars in our prompts.

Chain-of-Thought (CoT). CoT prompting decomposes translation into structured sub-steps, encouraging the model to reason through lexical, grammatical, and topical aspects before producing the final output (Wei et al., 2022). This approach is inspired by professional human translation workflows (Baker, 1992; Koehn, 2009; Bowker, 2002; Hatim and Munday, 2005).

Appendix A.2 presents the exact prompts used for the four strategies.

[1] https://ai.meta.com/

[2] https://deepmind.google/

[3] https://grok.com/

[4] https://deepseekv3.org/

4 Results and Discussion

4.1 RQ1: Benchmarking LLMs for Sylheti ⇔ Bangla

Figure 2 shows the BLEU score in both translation directions (Sylheti ⇔ Bangla). Across both directions, Grok 3 and LLaMA 4 are the strongest models, with LLaMA 4 leading Bangla→Sylheti (BLEU-1 = 0.3565; Grok 3 = 0.3525) and Grok 3 leading Sylheti → Bangla (0.4855; LLaMA 4 = 0.4656), while GPT-4.1 and Deepseek V3.2 trail on Bangla → Sylheti (both 0.2106). A pronounced directional asymmetry emerges: every model performs substantially better when translating into Bangla than into Sylheti—for the top systems, Grok 3 is 1.38 times higher (0.4855 vs. 0.3525) and LLaMA 4 is 1.31 times higher (0.4656 vs. 0.3565) on Sylheti → Bangla than Bangla → Sylheti, indicating that current LLMs are more proficient at producing the high-resource standard language than generating the dialect. This gap likely stems from pre-training data imbalance and limited exposure to Sylheti's lexicon, morphology, and orthography; as a result, models often normalize dialectal items into standard Bangla or omit Sylheti-specific function words. Qualitative examples in Table 1 show that zero-shot LLMs normalize Sylheti into standard Bangla, erasing dialectal lexicon, morphology, and particles. Core Sylheti words, e.g., ফুড়িটা (the girl), এখইন (now), যাইবা (will go), ফুয়াটায় (the boy), ফারলো (could/was able to), বাফর (father), মাই (mother), কেনিয়া (having bought), আনছইন (has brought), and আছইন নি? (is he not well?) are replaced by Bangla-leaning forms like মেয়াডা (the girl), এখন (now), যা (go), পোলাডা (the boy), পারল (could/was able to), আব্বার (father's), আম্মা (mother), কিনা (having bought), আনছে (has brought), and কেমন আছেগো? (how are you?). These errors reflect lexical substitution, morphological normalization (future, negation, honorifics), and orthographic drift, indicating limited Sylheti exposure and a decoding prior biased toward standard Bangla.

4.2 RQ2: Enhance LLM's Translation Performance by Sylheti-CAP

The evaluation of prompting strategies for both Bangla ⇔ Sylheti translation tasks across five LLMs shows a clear and consistent advantage for the proposed Sylheti-Context-Aware Prompting (Sylheti-CAP) method. Table 2 and 3 shows Sylheti-CAP achieves the highest scores across all models and evaluation metrics (BLEU-1, METEOR, and ChrF) in both translation directions.

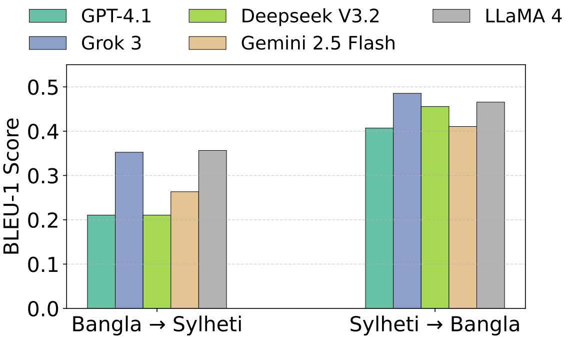


Figure 2: BLEU-1 scores on the test dataset for five LLMs (GPT-4.1, GPT-4.1-mini, LLaMA 4, Grok 3, and Deepseek V3.2) evaluated in both Bangla ⇔ Sylheti translation directions. BLEU scores are averaged over all test samples in each translation direction for this experiment.

As shown in Table 2, Sylheti-CAP consistently outperforms Zero-Shot, Few-Shot, and Chain-of-Thought (CoT) prompting. For example, Grok achieved the highest BLEU-1 (0.47) and ChrF (46.01), improving significantly over its Zero-Shot baseline (0.35 BLEU-1, 42.19 ChrF). LLaMA and GPT attained the top METEOR score of 0.34, while Deepseek's ChrF rose from 35.81 to 39.07 and Gemini improved from 0.26 to 0.29 BLEU-1. These gains highlight Sylheti-CAP's ability to inject dialect-specific context and structure into LLMs, enhancing lexical and semantic accuracy even in low-resource conditions.

Similarly, Table 3 demonstrates that Sylheti-CAP generalizes effectively in the reverse direction. Across all LLMs, it again delivers the best results for every metric. Deepseek reached the highest BLEU-1 (0.52), METEOR (0.45), and ChrF (53.72), outperforming its next-best CoT configuration. Grok followed closely with 0.52 BLEU-1 and 53.13 ChrF, while LLaMA, GPT, and Gemini also recorded steady improvements. For instance, LLaMA's ChrF increased from 49.95 to 52.19, and GPT's METEOR rose from 0.34 to 0.43, demonstrating Sylheti-CAP's consistent enhancement of lexical precision and semantic alignment. Across both translation directions, Sylheti-CAP proves highly effective in addressing data scarcity and capturing dialectal nuance. In contrast, Zero-Shot and CoT rely on general multilingual reasoning and lack sensitivity to dialectal variation. As shown in Table 4, the baseline

Table 1: Examples of Standard Bangla, Original Sylheti, and LLM-generated Sylheti translations produced by LLaMA 4 under the default prompt setting. Words highlighted in green represent accurate Sylheti translations corresponding to the Bangla source, while red highlights indicate errors or hallucinated terms generated by the model during Bangla→Sylheti translation.

Bangla Sentence	**Original Sylheti**	**Translated Sylheti**
তোমার আব্বু কেমন আছে? (How is your father?)	তোমার আব্বা বালা আছইন নি?	তোর আব্বা কেমন আছেগো?
মেয়েটি এখন সিলেট যাবে (The girl will now go to Sylhet)	ফুড়িটা এখইন সিলট যাইবা	মেয়াডা এখন সিলট যা
ছেলেটি আমাকে চিনতে পারল না (The boy could not recognize me)	ফুয়াটায় আমারে চিনতে ফারলো না	পোলাডা আমারে চিনবার পারল নাইব
আমার আব্বুর খুব গরম লাগছে (My father is feeling very hot)	আমার বাফর খুব গরম লাগছে	আমার আব্বার অনেক গরম লাগতাছে
আম্মু আমার জন্য নীল শাড়ি কিনে এনেছে(My mom bought a blue sari for me)	মাই আমার লাগি নীল শাড়ি কেনিয়া আনছইন	আম্মা আমার লাগি নীল শাড়ি কিনা আনছে

prompts frequently fall back to standard Bangla forms or non-target dialect words (e.g., বাপে (father), বাফা (father), আফা (older sister), পর (after), পরত (after/afterwards)), whereas Sylheti-CAP reliably selects the intended Sylheti lexicon such as আব্বা (father), ফরে (after), and বইনের (sister's). Only Sylheti-CAP yields a near-target variant, while other prompts produce lexically and morphologically off-target variants like বাপে কিরাম (how is your father) or কেমন আছের (how are you), and পর/পরত (after), but Sylheti-CAP correctly preserves core Sylheti dialect-specific word-to-word mappings for achieving lexically faithful Bangla→Sylheti translations.

4.3 Human Evaluation.

We conducted a human preference study on 200 samples for the Bangla ⇔ Sylheti translation task with 3 native speakers in Sylheti. Annotators rated translations from four prompting strategies Zero-Shot, Few-Shot, CoT, and Sylheti-CAP as Good, Fair, or Poor. Figure 3 shows that Sylheti-CAP consistently achieved the highest proportion of Good translations in both directions. For instance, Deepseek V3.2 and Grok 3 reached over 50% Good ratings in both Bangla → Sylheti and Sylheti → Bangla, while Poor outputs stayed below 20%. Overall, Sylheti-CAP substantially reduced low-quality outputs and increased human preference, confirming its effectiveness for dialect-aware translation.

4.4 LLM-as-a-judge.

We also conducted an LLM-as-a-judge study on the same set of 200 samples used in the human evaluation for the Bangla ⇔ Sylheti translation task. Using GPT-5.0, we directly scored adequacy, fluency, and overall translation quality on a 0–100 scale by comparing the reference Sylheti sentence with LLM-generated Sylheti translations under different prompting strategies. Appendix A.2 and Table 13 present the prompt used in the LLM-as-a-judge setup. Table 5 shows that Sylheti-CAP consistently achieves the highest adequacy, fluency, and overall scores, outperforming all other prompting strategies by a margin of 3–10 points.

4.5 MQM Evaluation.

To further analyze translation quality improvements across prompting strategies, we conducted Multidimensional Quality Metric (MQM) evaluations (Lommel, 2013) using the same 200 samples from the Bangla ⇔ Sylheti test sets. Following the expert-based annotation protocols in (Freitag et al., 2021; He et al., 2023), annotators identified translation errors, categorized them (e.g., omission, untranslated text, awkward phrasing, and mistranslation), and rated their severity. Each category contributed a weighted penalty, producing an overall MQM score per system.

As summarized in Table 6, Sylheti-CAP achieved the lowest (best) MQM scores in both directions (1.62 for Ben→Syl and 1.93 for Syl→Ben), outperforming Zero-Shot, Few-Shot, and CoT prompting. The category-level breakdown in Figure 4 shows that these improvements

Table 2: Translation performance (BLEU, ChrF, METEOR) of GPT-4.1, GPT-4.1-mini, LLaMA 4, Grok 3, and Deepseek V3.2 for ``Bangla→Sylheti" translation. Scores are the average of each test set for each language, measured using BLEU, ChrF, and METEOR metrics. Orange shading indicates that Sylheti-CAP outperformed other prompt strategies.

Model	**Zero-Shot**			**Few-Shot**			**COT**			**Sylheti-CAP**		
	B1	M	C	B1	M	C	B1	M	C	B1	M	C
Deepseek	0.21	0.24	35.81	0.10	0.07	14.47	0.27	0.19	35.38	0.32	0.24	39.07
Grok	0.35	0.28	42.19	0.39	0.27	41.57	0.33	0.26	39.81	0.47	0.30	46.01
LLaMA	0.36	0.26	37.09	0.35	0.32	42.22	0.34	0.25	38.23	0.42	0.34	45.08
GPT	0.36	0.32	42.68	0.32	0.30	43.34	0.34	0.29	40.60	0.42	0.34	43.91
Gemini	0.26	0.19	34.71	0.23	0.15	30.51	0.19	0.14	31.61	0.29	0.24	35.86

Table 3: Translation performance (BLEU, ChrF, METEOR) of GPT-4.1, GPT-4.1-mini, LLaMA 4, Grok 3, and Deepseek V3.2 for ``Sylheti→Bangla" translation. Scores are the average of each test set for each language, measured using BLEU, ChrF, and METEOR metrics. Blue shading indicates that Sylheti-CAP outperformed other prompt strategies..

Model	**Zero-Shot**			**Few-Shot**			**COT**			**Sylheti-CAP**		
	B1	M	C	B1	M	C	B1	M	C	B1	M	C
Deepseek	0.46	0.39	51.61	0.44	0.37	49.98	0.50	0.42	51.23	0.52	0.45	53.72
Grok 3	0.49	0.41	49.92	0.49	0.41	49.54	0.47	0.39	48.11	0.52	0.44	53.13
LLaMA	0.47	0.41	49.95	0.45	0.37	47.73	0.45	0.39	51.01	0.49	0.41	52.19
GPT	0.41	0.34	46.45	0.50	0.40	48.35	0.41	0.33	44.09	0.47	0.43	51.49
Gemini	0.41	0.34	46.72	0.41	0.34	46.83	0.40	0.33	45.95	0.46	0.39	48.69

are primarily driven by reductions in mistranslations, awkward phrasing, and omission errors, where Sylheti-CAP consistently yields lower penalties (e.g., 580 vs. 670 for mistranslation and 200 vs. 220 for omission compared to Zero-Shot). These findings indicate that incorporating dialectal context and linguistic grounding not only reduces literal translation errors but also enhances overall fluency and semantic adequacy.

4.6 LLMs' Hallucinations.

In natural language generation (NLG), hallucination refers to the production of content that is nonsensical or unfaithful to the source text (Filippova, 2020; Zhang et al., 2019), and remains a persistent challenge for LLMs (Zhang et al., 2023). To examine this issue within the context of Bangla ⇔ Sylheti translation, we conducted a human evaluation of hallucination errors across four prompting strategies. Using 200 sampled sentences from each translation direction, annotators inspected the generated outputs from five LLMs and labeled whether each contained hallucinated or semantically inconsistent content, following the definition in (Guerreiro et al., 2023).

As illustrated in Figure 5, Sylheti-CAP consistently achieves the lowest hallucination rates across all models (e.g., 12.6–13.8%), outperforming CoT, Few-Shot, and Zero-Shot prompting, which exhibit higher rates (typically 15–17%). We attribute this reduction to the contextual grounding of Sylheti-CAP, which integrates dialect-specific translation cues and semantic constraints directly into the prompt. This additional linguistic guidance helps steer the model's token generation away from spurious continuations, improving overall faithfulness and reducing nonsensical or unaligned outputs.

5 Related Works

LLMs in Machine Translation. Recent advances in LLMs such as GPT-4 (OpenAI et al., 2023) and LLaMA (Touvron et al., 2023) have significantly advanced Neural Machine Translation (NMT) (Jiao et al., 2023; Hendy et al., 2023). Two main paradigms dominate: in-context learning (ICL) and fine-tuning. ICL enables LLMs to perform translation tasks from a few exemplars

Table 4: Examples of Standard Bangla, Original Sylheti, and LLM-generated Sylheti translations produced by LLaMA 4 under the Zero-Shot Few-Shot COT Sylheti-CAP prompt setting. Words highlighted in green represent accurate Sylheti translations corresponding to the Bangla source, while red highlights indicate errors or hallucinated terms generated by the model during Bangla→Sylheti translation.

Bangla	**Sylheti**	**Zero-Shot**	**Few-Shot**	**COT**	**Sylheti-CAP**
তোমার আব্বু কেমন আছে?	তোমার আব্বা বালা আছইন নি?	তোমার বাপে কিরাম আছে?	তোমার আব্বা কেমন আছের?	তুমার বাফা ক্যামন আছইন?	তুমার আব্বা বালা আছইন?
আমার দুইদিন পরে বিয়ে হবে	আমার দুইদিন ফরে বিয়া অইবো	আমার দুই দিন পরত বিয়া অইব	আমার দুই দিন পর বিয়া অইবো	আমার দুই দিন পর বিয়া অইব	আমার দুই দিন ফরে বিয়া অইবো
আমার বড় বোনের আজকে মন ভালো নেই	আমার বড় বইনর আইজ মন ভালা নায়	আমার বড় আফা অহন মন বালা নাই	আমার বড় আফার আইজকু মন ভালা নায়	আমার বড় আফার আজকা মন বালা নাই	আমার বড় বইনের আইজকু মন ভালা নায়

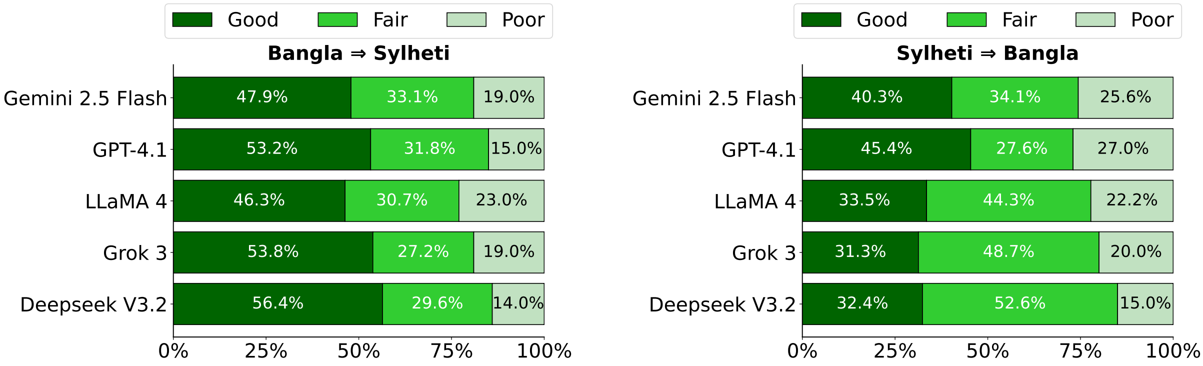


Figure 3: Human preference study comparing Sylheti-CAP with Zero-Shot, Few-Shot and COT for LLMs (GPT-4.1, GPT-4.1-mini, LLaMA 4, Grok 3, and Deepseek V3.2).

Table 5: GPT-5.1-as-a-judge average scores (0–100) for adequacy, fluency, and overall translation quality, comparing reference Sylheti sentences with LLM-generated Sylheti translations across different prompting strategies.

Prompt	**Adequacy**	**Fluency**	**Overall**
Zero-shot	72.7	77.5	75.6
Few-shot	78.5	79.5	82.4
CoT	76.3	78.2	78.8
Sylheti-CAP	**84.2**	**86.5**	**85.3**

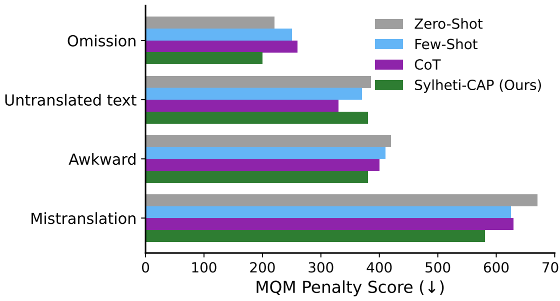


Figure 4: MQM penalty scores across different error categories for 200 test sentences from each of the Bangla⇔Sylheti test sets. Lower scores is less severe translation errors.

without parameter updates (Brown et al., 2020), often matching supervised models (García et al., 2023). The quality of demonstrations strongly influences performance (Agrawal et al., 2022). In contrast, fine-tuned models such as XGLM-7B (Li et al., 2023) and instruction-tuned variants (Chen et al., 2021) improve translation faithfulness and low-resource adaptability.

Evaluation of LLM-based translation generally follows two directions: (1) Prompt-level design, focusing on prompt templates, demonstration selection, and reasoning structure (Vilar et al., 2022; Zhang et al., 2023; Jiao et al., 2023); and (2) Comprehensive benchmarking, testing multilingual (Hendy et al., 2023; Zhu et al., 2023), document-level (Karpinska and Iyyer, 2023), low-resource (García et al., 2023), and hallucination-resistant (Guerreiro et al., 2023)

Table 6: Averaged MQM scores (↓) for different prompting strategies on Bangla–Sylheti (Ben→Syl) and Sylheti–Bangla (Syl→Ben) translation tasks. Lower values indicate fewer translation errors and better quality.

Prompt	**Ben→Syl**	**Syl→Ben**
Zero-Shot	2.54	3.02
Few-Shot	2.41	2.87
CoT	2.18	2.56
Sylheti-CAP	**1.62**	**1.93**

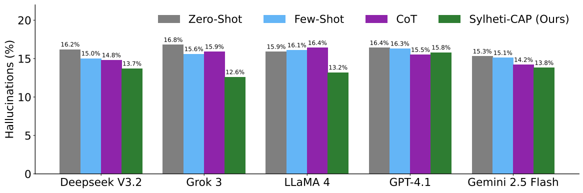


Figure 5: Ratio of hallucinations in generated translations for 200 test sentences from each of the English⇔Bengali test sets. Human annotators labeled each output as either containing or not containing a hallucination error.

settings, often incorporating human feedback (Jiao et al., 2023). While early efforts to use cross-sentence context showed limited gains (Lopes et al., 2020; Fernandes et al., 2021), recent LLMs can dynamically leverage document-level and contextual cues (Karpinska and Iyyer, 2023; Wang et al., 2023). Newer methods integrate retrieval-based prompting (Agrawal et al., 2022), bilingual lexicons (Ghazvininejad et al., 2023), context-aware prompting (Pilault et al., 2023), and document-level fine-tuning (Wu et al., 2024). However, LLMs' potential to fully exploit bilingual, multi-turn contextual signals and context-aware evaluation remains underexplored—particularly for low-resource and dialectal translation, where context injection can close significant linguistic gaps.

Bangla Machine Translation. Early MT efforts for Bangla concentrated on the high-resource Bangla–English pair. For Bangla → English, studies have employed Sequence-to-Sequence (Seq-to-Seq) models utilizing attention-based Recurrent Neural Networks (RNNs) (Islam et al., 2023). Conversely, English → Bangla translation has been successfully achieved using encoder–decoder Gated Recurrent Unit (GRU) architectures, which were shown to outperform LSTM-based models (Mahmud et al., 2021). Beyond specific models, comprehensive analyses have benchmarked multiple NMT architectures for the general Bangla–English task (Hasan et al., 2019). More recent work has leveraged transformer-based models with large-scale multi-dialect parallel corpora to address generalized dialectal Bangla translation (Faria et al., 2023). Addressing dialectal variation, efforts on the Chittagonian dialect have applied rule-based morphological transformations and bidirectional mappings for conversion (Milon et al., 2020; Hossain et al., 2022). For Sylheti, foundational work has provided essential grammatical insights (Goswami, 2021). In NMT, a Sylheti → Bangla system was previously introduced using a BiLSTM (Prama and Anwar, 2025b) and transformer based architecture (Oni and Prama, 2025). Despite these contributions, the Bangla–Sylheti pair remains significantly underexplored due to scarce standardized corpora, substantial orthographic variation, and limited linguistic resources. To the best of our knowledge, this study is the first to employ and systematically evaluate Large Language Models (LLMs) for the challenging Bangla ⇔ Sylheti dialect translation task.

6 Conclusion

This study presents the first systematic evaluation of Large Language Models (LLMs) for Bangla–Sylheti Machine Translation. We propose Sylheti-CAP (Context-Aware Prompting), a framework that integrates linguistic rules, bilingual dictionaries, and contextual fluency constraints directly into prompts to generate accurate and natural Sylheti translations. Experiments across five advanced LLMs (GPT-4.1, GPT-4.1-mini, LLaMA 4, Grok 3, and Deepseek V3.2) show that Grok 3 and LLaMA 4 achieve the highest BLEU and METEOR scores in both translation directions. Sylheti-CAP consistently outperforms zero-shot, few-shot, and chain-of-thought baselines, reducing hallucinations, mistranslations, and awkward phrasing. Overall, Sylheti-CAP demonstrates a scalable, linguistically grounded approach for low-resource and dialectal translation, paving the way for improved translation quality across other Bangla dialects and underrepresented languages.

7 Limitations

While Sylheti-CAP demonstrates significant improvements in Bangla–Sylheti translation, several limitations remain. The framework relies solely on prompting without model fine-tuning. Incorporating fine-tuned word embeddings could provide a more stable and permanent improvement in translation performance. Current bilingual dictionary consists of only 2260 word pairs expanding it to include a wider range of dialect-specific and context-rich words would likely enhance translation quality and coverage. Prominent LLMs used in this study—such as GPT-4.1, LLaMA 4, and Deepseek V3.2 are primarily trained on data from high-resource languages. Since their pretraining corpora likely contain limited or no Sylheti text, this lack of exposure may constrain their dialectal understanding. Moreover, the absence of publicly available training data for proprietary models limits reproducibility and transparency. Finally, our human evaluation involved a small number of native Sylheti speakers from different regions. Although care was taken to ensure linguistic proficiency and regional diversity, subjective variation remains, and the results may not fully generalize. Conducting broader evaluations with more participants and developing standardized Sylheti evaluation datasets would strengthen benchmarking and comparability in future work.

References

Accredited Language Services. 2015. Bengali. Accessed: 15 Sept. 2015.

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2022. In-context examples selection for machine translation. In *Annual Meeting of the Association for Computational Linguistics*.

Meta AI. 2024. Introducing llama 4: Advancing multimodal intelligence.

Duarte M. Alves, Nuno M. Guerreiro, João Alves, José P. Pombal, Ricardo Rei, José G. C. de Souza, Pierre Colombo, and André Martins. 2023. Steering large language models for machine translation with finetuning and in-context learning. In *Conference on Empirical Methods in Natural Language Processing*.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. *ArXiv*, abs/1606.02006.

Mona Baker. 1992. In other words: A coursebook on translation.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*.

Lynne Bowker. 2002. Computer-aided translation technology: A practical introduction.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Sheila Castilho and Rebecca Knowles. 2024. A survey of context in neural machine translation and its evaluation. *Natural Language Processing*.

Rajen Chatterjee, Markus Freitag, Matteo Negri, and Marco Turchi. 2020. Findings of the wmt 2020 shared task on automatic post-editing. In *Conference on Machine Translation*.

Guanhua Chen, Shuming Ma, Yun Chen, Dongdong Zhang, Jia-Yu Pan, Wenping Wang, and Furu Wei. 2021. Towards making the most of cross-lingual transfer for zero-shot neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*.

Google DeepMind. 2025. Gemini 2.5 flash. `https://deepmind.google/models/gemini/flash/`. Accessed: 2025-10-04.

DeepSeek-AI. 2024. Deepseek-v3 technical report. `https://arxiv.org/abs/2412.19437`. Accessed: 2025-10-04.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2022. A survey on in-context learning. In *Conference on Empirical Methods in Natural Language Processing*.

Darshil Doshi, Aritra Das, Tianyu He, and Andrey Gromov. 2023. To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets. *ArXiv*, abs/2310.13061.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Fatema Tuj Johora Faria, Mukaffi Bin Moin, Ahmed Al Wase, Mehidi Ahmmed, Md. Rabius Sani, and Tashreef Muhammad. 2023. Vashantor: A large-scale multilingual benchmark dataset for automated translation of bangla regional dialects to bangla language. *ArXiv*, abs/2311.11142.

Patrick Fernandes, Kayo Yin, Graham Neubig, and André F. T. Martins. 2021. Measuring and increasing context usage in context-aware machine translation. *ArXiv*, abs/2105.03482.

Katja Filippova. 2020. Controlled hallucinations: Learning to generate faithfully from noisy data. In *Findings*.

Markus Freitag, George F. Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460--1474.

Xavier García, Yamini Bansal, Colin Cherry, George F. Foster, Maxim Krikun, Fan Feng, Melvin Johnson, and Orhan Firat. 2023. The unreasonable effectiveness of few-shot learning for machine translation. *ArXiv*, abs/2302.01398.

Marjan Ghazvininejad, Hila Gonen, and Luke Zettlemoyer. 2023. Dictionary-based phrase-level prompting of large language models for machine translation. *ArXiv*, abs/2302.07856.

A. Goswami. 2021. Marked geminates as evidence of sonorants in sylheti bangla: An optimality account. *Acta Linguistica Asiatica*, 11(1):99--112.

Nuno M. Guerreiro, Duarte M. Alves, Jonas Waldendorf, Barry Haddow, Alexandra Birch, Pierre Colombo, and André Martins. 2023. Hallucinations in large multilingual translation models. *Transactions of the Association for Computational Linguistics*, 11:1500--1517.

Mika Hämäläinen and Khalid Alnajjar. 2019. A template based approach for training nmt for low-resource uralic languages - a pilot with finnish. *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*.

M. A. Hasan, F. Alam, S. A. Chowdhury, and N. Khan. 2019. Neural machine translation for the bangla-english language pair. In *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, pages 1--6, Dhaka, Bangladesh.

Basil A. Hatim and Jeremy Munday. 2005. Translation: An advanced resource book.

Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. 2023. Exploring human-like translation strategy with large language models. *Transactions of the Association for Computational Linguistics*, 12:229--246.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *ArXiv*, abs/2302.09210.

Nahid Hossain, Hafizur Rahman Milon, Sheikh Nasir Uddin Sabbir, and Azfar Inan. 2022. Inclusive bidirectional conversion system between chittagonian and standard bangla. *Bulletin of Electrical Engineering and Informatics*.

Tsuneo Inaba, Kenji Tsuchida, Tadahiko Sugibayashi, Shuichi Tahara, and Hiroaki Yoda. 2003. Resistance ratio read (r/sup 3/) architecture for a burst operated 1.5v mram macro. *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.*, pages 399--402.

R. Islam, M. Hasan, M. Rashid, and R. Khatun. 2023. Bangla to english translation using sequence to sequence learning model based recurrent neural networks. In *Machine Intelligence and Emerging Technologies (MIET 2022)*, volume 490 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 481--494. Springer, Cham.

Wenxiang Jiao, Wenxuan Wang, Jen-Tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *ArXiv*, abs/1607.01759.

Marzena Karpinska and Mohit Iyyer. 2023. Large language models effectively leverage document-level context for literary translation, but critical errors persist. In *Conference on Machine Translation*.

Philipp Koehn. 2009. A process study of computer-aided translation. *Machine Translation*, 23:241--263.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *NMT@ACL*.

Jiahuan Li, Hao Zhou, Shujian Huang, Shan Chen, and Jiajun Chen. 2023. Eliciting the translation ability of large language models via multilingual finetuning with translation instructions. *Transactions of the Association for Computational Linguistics*, 12:576--592.

Arle Lommel. 2013. Multidimensional quality metrics : A flexible system for assessing translation quality.

António V. Lopes, M. Amin Farajian, Rachel Bawden, Michael J.Q. Zhang, and André F. T. Martins. 2020. Document-level neural mt: A systematic comparison. In *European Association for Machine Translation Conferences/Workshops*.

A. Mahmud, M. M. Al Barat, and S. Kamruzzaman. 2021. Gru-based encoder-decoder attention model for english to bangla translation on novel dataset. In *2021 5th International Conference on Electrical Information and Communication Technology (EICT)*, pages 1--6, Khulna, Bangladesh.

Sameen Maruf, Fahimeh Saleh, and Gholamreza Haffari. 2019. A survey on document-level machine translation: Methods and evaluation. *ArXiv*, abs/1912.08494.

Hafizur Rahman Milon, Sheikh Nasir Uddin Sabbir, Azfar Inan, and Nahid Hossain. 2020. A comprehensive dialect conversion approach from chittagonian to standard bangla. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 214--217.

Mangsura Kabir Oni and Tabia Tanzin Prama. 2025. Transformer-based low-resource language translation: A study on standard bengali to sylheti.

OpenAI. 2024. Introducing gpt-4.1. `https://openai.com/index/introducing-gpt-4-1/`. Accessed: 2025-10-04.

Josh Achiam OpenAI, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, and 260 others. 2023. Gpt-4 technical report.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics*.

Keqin Peng, Liang Ding, Qihuang Zhong, Li Shen, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2023. Towards making the most of chatgpt for machine translation. In *Conference on Empirical Methods in Natural Language Processing*.

Jonathan Pilault, Xavier García, Arthur Bravzinskas, and Orhan Firat. 2023. Interactive-chain-prompting: Ambiguity resolution for crosslingual conditional generation with interaction. In *International Joint Conference on Natural Language Processing*.

Maja Popovic. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *WMT@EMNLP*.

Tabia Tanzin Prama and Md Musfique Anwar. 2025a. Sylheti to standard bangla neural machine translation: A deep learning-based dialect conversion approach. In Akhilesh Bajaj, Ana Maria Madureira, and Ajith Abraham, editors, *Hybrid Intelligent Systems. HIS 2023. Lecture Notes in Networks and Systems*, volume 1224. Springer, Cham.

Tabia Tanzin Prama and Md Musfique Anwar. 2025b. Sylheti to standard bangla neural machine translation: A deep learning-based dialect conversion approach. In *Hybrid Intelligent Systems*, volume 1224 of *Lecture Notes in Networks and Systems*, pages 208--217, Cham. Springer. Presented at HIS 2023, published online 27 July 2025.

Tabia Tanzin Prama, Christopher M. Danforth, and Peter Sheridan Dodds. 2025. Banglamath : A bangla benchmark dataset for testing llm mathematical reasoning at grades 6, 7, and 8. *ArXiv*, abs/2510.12836.

Tabia Tanzin Prama and Mangsura Kabir Oni. 2025. A dataset for translating local bangla (sylheti) dialects into standard bangla.

Vikas Raunak, Arul Menezes, Matt Post, and Hany Hassan Awadallah. 2023. Do gpts produce less literal translations? *ArXiv*, abs/2305.16806.

Nathaniel Romney Robinson, Perez Ogayo, David R. Mortensen, and Graham Neubig. 2023. Chatgpt mt: Competitive for high- (but not low-) resource languages. *ArXiv*, abs/2309.07423.

Candide Simard, Sarah M. Dopierala, and E. Marie Thaut. 2020. Introducing the sylheti language and its speakers, and the soas sylheti project.

Nusrat Sultana, Rumana Yasmin, Bijon Mallik, and Mohammad Shorif Uddin. 2025. Onubad: A comprehensive dataset for automated conversion of bangla regional dialects into standard bengali dialect. *Data in Brief*, 58.

Nllb team, Marta Ruiz Costa-jussà, James Cross, Onur cCelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Alison Youngblood, Bapi Akula, Loïc Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2022. No language left behind: Scaling human-centered machine translation. *ArXiv*, abs/2207.04672.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George F. Foster. 2022. Prompting palm for translation: Assessing strategies and performance. *ArXiv*, abs/2211.09102.

Siyin Wang, Chao-Han Huck Yang, Ji Wu, and Chao Zhang. 2023. Can whisper perform speech-based in-context learning? *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13421--13425.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Minghao Wu, Thuy-Trang Vu, Lizhen Qu, George Foster, and Gholamreza Haffari. 2024. Adapting large language models for document-level machine translation. *ArXiv*, abs/2401.06468.

xAI. 2025. Grok 3 beta — the age of reasoning agents. https://x.ai/news/grok-3. Accessed: 2025-10-04.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *ArXiv*, abs/2409.12122.

Kayo Yin, Patrick Fernandes, André F. T. Martins, and Graham Neubig. 2021. When does translation require context? a data-driven, multilingual exploration. In *Annual Meeting of the Association for Computational Linguistics*.

Jiajun Zhang and Chengqing Zong. 2016. Bridging neural machine translation and bilingual dictionaries. *ArXiv*, abs/1610.07272.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. *ArXiv*, abs/1906.02448.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alexander J. Smola. 2023. Multimodal chain-of-thought reasoning in language models. *Trans. Mach. Learn. Res.*, 2024.

Xing Jie Zhong and David Chiang. 2020. Look it up: Bilingual and monolingual dictionaries improve neural machine translation. *ArXiv*, abs/2010.05997.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Lingpeng Kong, Jiajun Chen, Lei Li, and Shujian Huang. 2023. Multilingual machine translation with large language models: Empirical results and analysis. *ArXiv*, abs/2304.04675.

Table 7: Examples from the Bangla⇔Sylheti word-to-word dictionary.

Bangla	**Sylheti**
মহিলার	বেটির
হবে	অইবো
উপরে	উফরে
একটাই	এখটাউ
একেক	এখনত
এলাকার	জাগার
রকম	লাখান
নতুনরা	নয়া
অনেক	বহুততা
শিখতে	হিকতা
করি	খরি
হোকনা	অউক
কিছুর	কুন্তার
শুরুটা	শুরু
এভাবেই	অলাউ
আমার	মোর
সাথে	লগে
কথা	মাতবায়
অনেক	বাক্কা
হবে	লাগবো
সবার	হখলর
কত	খত

A Appendix

A.1 Bangla⇔Sylheti Dictionary

To build a comprehensive bilingual lexicon, we merged three parallel corpora: Vashantor (Faria et al., 2023) (2,125 sentences), ONUBAD (Sultana et al., 2025) (980 sentences), and the Sylheti Dataset (Prama and Oni, 2025) (5,002 sentence pairs). Since Sylheti is a dialect of Bangla, a large portion of the vocabulary overlaps between the two. However, there are also numerous dialect-specific variations in phonology, morphology, and semantics. From these datasets, we compiled a word-to-word dictionary containing 2260 aligned sentence pairs, focusing on words and expressions unique to Sylheti. Here is the dictionary of Bangla⇔Sylheti Dictionary: https://github.com/word mapping 2260.csv. Table 7 shows the Bangla⇔Sylheti word-to-word dictionary.

A.2 Prompt Strategies

Table 8: Examples of identical words in Bangla and Sylheti. While Sylheti is a dialect of Bangla, many words remain unchanged due to shared linguistic roots, phonetic overlap, and common Indo-Aryan origin. These lexical similarities contribute to overall translation fluency between the two languages.

Bangla	**Sylheti**
তুমি	তুমি
রাজকুমারির	রাজকুমারির
মায়া	মায়া
জীবন	জীবন
রঙিন	রঙিন
ছবি	ছবি
আর	আর
আশেপাশে	আশেপাশে

Table 9: Zero-Shot Prompt: Direct instruction for Bangla→Sylheti translation without examples or prior context.

Prompt:

You are a professional translator proficient in both Bangla and Sylheti. Your task is to translate the following Bangla sentence into natural and fluent Sylheti. Provide only the translated Sylheti sentence without any additional explanation.
Bangla: "<input_sentence>"
Sylheti:

Table 10: Few-Shot Prompt: Translation prompt with six Bangla–Sylheti example pairs to guide model behavior.

Prompt:

You are given Bangla sentences and asked to translate them into Sylheti. Here are a few examples:
Bangla–Sylheti Examples:
১. কেমন আছো ? → ভালা আছনি?
২. আজকে আমার মন ভালো নেই → আইজকু আমার মন ভালা নায়
৩. তুমি কি করো ? → তুমি কিতা খরো?
৪. এই গরমে আমার কিছু ভালো লাগে না → অউ গরমো আমার কুনতা ভালা লাগের না
৫. ছেলেটি সাদা রঙয়ের একটি শার্ট পরে এসেছিল → ফুয়াটায় এখটা সাদা রংগর শার্ট পিন্দিয়া আইছিল

Instruction: Translate the following Bangla sentence into Sylheti:
Bangla: "<input_sentence>"
Sylheti:

Table 11: Chain-of-Thought (CoT) Prompt: A structured, reasoning-based prompt for multi-step contextual translation.

Prompt:

You are a translation assistant that follows a three-step process: **Knowledge Mining → Knowledge Integration → Knowledge Selection.** Your goal is to translate the given Bangla text into Sylheti as accurately and fluently as possible.

Step 1: Knowledge Mining 1. Extract the keywords from the input **Bangla** sentence and translate them into Sylheti. *Output:* Keyword Pairs: <src_word1>:<tgt_word1>, ...

2. Identify a few words describing the main topics of the sentence. *Output:* Topics: <topic1>, <topic2>, ...

3. Write a **Bangla** sentence related to but different from the input, and provide its **Sylheti** translation. *Output:* <src_demo> | <tgt_demo>

Step 2: Knowledge Integration Combine the mined knowledge to generate a candidate translation.

Prompt: Keyword Pairs: ...
Topics: ...
Related Example: <src_demo> | <tgt_demo>

Instruction: Given the above, translate the following **Bangla** sentence into **Sylheti**.
Bangla: "`<input_sentence>`"
Sylheti: <Candidate Translation>

Step 3: Knowledge Selection Compare all candidate outputs (Keyword, Topic, Demo, Base) and select the most fluent and accurate final translation.
Output: Best Translation: <final_output>

Table 12: Sylheti-CAP Prompt: Context-Aware Prompt integrating explicit linguistic rules and word mappings for authentic Bangla→Sylheti translation.

Prompt:

You are a translator specializing in **Sylheti**, a distinct Indo-Aryan language closely related to Bangla but with its own grammar, vocabulary, and phonology. Your task is to translate Bangla sentences into natural, fluent Sylheti speech while preserving meaning, grammar, and idiomatic usage. Follow all the rules and mapping guidelines below when producing the translation.
□□ **Grammar and Pronouns:**

- Replace Bangla pronouns with Sylheti equivalents: আমি → মুই, তুমি → তুমি/তুই, আপনি → আফনে, আমরা → আমরার, তারা → তারার, সে → হে/তাই.
- For possessives: আমার → মোর, তোমার → তুমার, আমাদের → আমরার, আপনাদের → আফনারার.

□□ **Questions:** Use Sylheti interrogatives. কী → কিতা, কোথায় → কুনান/কুনানো, কেমন → কিলা, কেন → কিয়েন, কত → কিত্তা.
□□ **Verbal Rules:**

- Drop aspiration: খ → ক, ঘুম → গুম.
- Present tense endings: আমি করি → মুই খরি, তুমি করো → তুমি খরো, সে করে → হে খরে.
- Past tense: করেছিলাম → খরসিলাম.
- Future tense: করবো → খরমু.
- Negation: না → নি / নায়. Example: আমি যাই না → মুই যাই নি.
- Copula: আছে / আছি / আছো → রইছে / আছি / আসো.

□□ **Vocabulary:** পড়াশোনা → পড়ালেখা, টাকা → ফইশা, বন্ধু → বন্দু, বাড়ি → গর, খুশি → কুশি, দুঃখ → বেজার.
□□ **Imperatives:** খাও → খা / খাইওকা (polite), বসো → বইবা, যাও → যা.

□□ **Passive Voice:** জানালা ছেলেটা ভেঙেছে → জানালা ফুয়া ডি বাঙ্গা অইসে. Pattern: *Object + Subject + dia + participle + oisil/oise/or.*
□□ **Classifiers:** একটা → এখটা, পাঁচটা → ফাসটা.

□□ **Syntactic and Morphological Directives:** Always preserve the SOV (Subject–Object–Verb) order. Modify pronouns, verbs, negations, and key vocabulary to reflect Sylheti tone and grammar. Output must sound like spoken Sylheti, not formal Bangla.
□□ **Reference Word Mapping Dictionary (Excerpt):** Use the following word-level mappings when applicable: মহিলার → বেটির, হবে → অইবো, এলাকার → জাগার, শিখতে → হিকতা, করি → খরি, ভালো → ভালা, সাথে → লগে, কথা → মাতবায়, ছবি → ছবি, যাবে → যাইবো, কিছু → কুনতা, আমার → মোর, আপনি → আফনে.
Final Instruction: Translate the following Bangla text into fluent Sylheti, adhering to all rules and mappings above. Ensure the translation reflects natural spoken Sylheti and not literal Bangla.
Bangla: `"<input_sentence>"`
Sylheti:

Table 13: Sylheti-CAP Prompt: Context-Aware Prompt integrating explicit linguistic rules and word mappings for authentic Bangla→Sylheti translation.

Prompt:

LLM-as-a-judge prompt
You are an expert bilingual evaluator.
Your task is to evaluate a MACHINE TRANSLATION from Standard Bangla to Sylheti.
SOURCE (Standard Bangla): <*SOURCE SENTENCE*>
REFERENCE TRANSLATION (Sylheti): <*REFERENCE TRANSLATION*>
CANDIDATE TRANSLATION (Translated Sylheti): <*CANDIDATE TRANSLATION using different prompt strategy*>
Please rate the candidate translation on a scale from 0 to 100 for:
1. ADEQUACY: how well it preserves the meaning of the source.
2. FLUENCY: how natural and grammatically correct the text is in Sylheti.
3. OVERALL: your overall judgment of translation quality.
Return your answer in JSON format ONLY, as:
`{"adequacy": X, "fluency": Y, "overall": Z}`

Clustering LLM-based Word Embeddings to Determine Topics from Bangla Articles

Rifat Rahman
Department of CSE & IAT
Bangladesh University of Engineering
& Technology
rifatrahman05007@gmail.com

Mohammed Eunus Ali
Department of CSE
Bangladesh University of Engineering
& Technology
mohammed.eunus.ali@gmail.com

Abstract

Topic modeling methods identify fundamental themes within textual documents, facilitating an understanding of the insights inside them. Traditional topic modeling approaches are based on the generative probabilistic process that assumes the document-topic and topic-word distribution. Hence, those approaches fail to capture semantic similarities among words inside the documents and are less scalable with the vast number of topics and documents. This paper presents a method for capturing topics from Bangla documents by clustering the word vectors induced from LLM models. Corpus statistics are integrated into the clustering & word reordering process within each cluster or topic to extract the top words. Additionally, we deploy dimensionality reduction techniques, such as PCA, prior to clustering. Finally, we perform a comparative study and identify the best-performing combination of clustering and word embedding methods. Our top-performing combination outperforms the traditional probabilistic topic model in capturing topics and top words per topic, and excels notably in terms of computational efficiency and time complexity.

1 Introduction

Topic modeling is a data analysis technique highly used for text mining in Natural Language Processing (NLP) (Sia et al., 2020). Topic models discover the key themes and patterns from a large corpus of textual documents by analyzing and grouping words into clusters or topics based on their co-occurrences inside the documents (Boyd-Graber et al., 2017). Thus, it helps analyze big data, capturing subjects discussed in the text. Topic modeling has several usabilities in NLP, like feature engineering (Rahman, 2020b), sentiment analysis (Rahman et al., 2022), document categorization (Rahman, 2020a), etc. In this study, we present a benchmark investigation focused on the weighted clustering of LLM-based word embeddings to extract topics from Bangla language documents. Additionally, we undertake a comparative analysis, directly comparing our top-performing approach with the widely adopted topic modeling technique, Latent Dirichlet Allocation (LDA).(Blei et al., 2003).

Word embedding refers to the vector representations of words in multi-dimensional space that keep the semantic characteristics of words (Rahman, 2020b) within the corpus. As it holds the semantic attributes of words, similar words stay closely in the multi-dimensional space. Hence, clustering those vectors gives valuable insights related to the main themes of documents. Again, pretrained LLM models (Wang et al., 2019) consider the attention mechanism that enables the model to effectively hold long-range dependencies among words or phrases in texts. So, we apply clustering on LLM-based word embeddings to identify the topics as clusters of words.

The majority of studies apply probabilistic approaches (Blei et al., 2010) (e.g., LDA, pLSA, biterm topic model, etc.) or linear algebra-based techniques (e.g., LSA). Probabilistic or linear algebra-based topic models do not consider linguistic features inside the corpus. Very few works (De Miranda et al., 2020; Sridhar, 2015; Sia et al., 2020) that conduct clustering on word embeddings do not take standardized word vectors or LLM-based word embeddings into account. Most notably, clustering word embeddings has not yet been explored for topic modeling in the context of the Bangla language.

Our *objective* of this work is to propose a topic model technique by identifying the best-performing combination of clustering algorithm and LLM-based word embeddings that outperforms traditional probabilistic topic models.

To achieve our goal, we introduce a novel topic modeling technique in Bangla by clustering LLM-based word embedding methods. We apply centroid-based weighted clustering as centroid-

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 309–321
December 23, 2025 ©2025 Association for Computational Linguistics

based clustering helps identify the top words of individual clusters based on the distance from the cluster centers. Weighted clustering is done by incorporating statistical information from the corpus. We utilize LLM-based pre-trained word embeddings as those models have been trained on the vast amount of various contextual information. So, those word embeddings can be regarded as standard embeddings. The dimensionality reduction algorithm, Principle Component Analysis (PCA) (Wold et al., 1987), is applied to word embeddings before clustering. After identifying the clustered words in each cluster, we reorder words according to the frequency statistics of words within the documents to find the top X words. We also do comparison among various combinations of LLM-based word embeddings & clustering methods and find the best-performing combination. The best-performing combination is then compared against LDA. In the meantime, we explore the word embedding technique that performs comparatively well with all clustering processes and the clustering method that gives the best result for all types of vector representations. Our best-performing combination (average NPMI score=0.31) extracts top words having more point-wise mutual information or coherence within a topic than LDA (average NPMI score=0.18). Moreover, our approach requires less run time and computational power than LDA.

The *contribution* of our study is threefold:

- We propose the first word-embedding & clustering based topic modeling for Bangla.
- We incorporate document statistics and linguistic features in our topic models that help extract more informative topic words.
- We conduct a benchmark study and identify the best word embedding technique for clustering and the best clustering method for all kinds of word embeddings.

2 Related Works

Clustering is a much-used approach for analyzing documents and texts. A plethora of studies apply clustering on texts for readability measurement (Cha et al., 2017), argument identification from texts (Reimers et al., 2019), and text classification tasks (Sato et al., 2017). Cha et al. (2017) apply clustering word embeddings for predicting text readability and show how the approach improves overall performance and the text is suitable for the intended reader's comprehension level. They also perform sentence matching based on semantic similarity by clustering word embeddings. Another study (Sato et al., 2017) utilizes the clustering method on paragraph vectors to capture semantic similarities among documents and phrases that outperforms the co-embedding method utilizing bag-of-words representation. Again, Reimers et al. (2019) explore the effectiveness of two contemporary contextualized word embedding techniques, ELMo and BERT, for argument-searching tasks. These techniques are used to classify and cluster arguments specific to various topics. However, clustering word embedding has little been explored regarding topic modeling.

Several studies attempt to include word embeddings in probabilistic topic modeling. Liu et al. (2015) introduce topical word embeddings (TWE) for creating multi-prototype word embeddings where word vectors are different based on topics. They use LDA to determine word topics and apply collapsed Gibbs sampling to assign topics to each word token. Another research work (Nguyen et al., 2015) develops a hybrid version of the topic modeling method, expanding two different probabilistic topic models by integrating word embeddings trained on a little data to figure out the word-topic distribution. These models achieve notable improvements in topic coherence, document clustering, and document classification. Das et al. (2015) introduce an alternative parameterization of "topics" in the LDA framework where topics are represented as categorical distributions over concealed word types, combined with multivariate Gaussian distributions in the embedding space. Some authors (Zhao et al., 2017) present the WEI-FTM that yields focused topics with representative words, enhancing perplexity and topic quality. It efficiently employs a Gibbs sampling algorithm for inference, accommodating both regular and short texts without loss of generality. Dieng et al. (2020) implement the Embedded Topic Model (ETM), merging probabilistic generative topic models with word vectors that outperform LDA for identifying topics from short-sized texts or documents. The model also shows robustness with larger vocabularies.

Very few works have investigated the efficacy of directly clustering word embeddings for topic analysis. Xie and Xing (2013) propose a Multi-

Grain Clustering Topic Model (MGCTM) that clusters similar documents and introduces topics for those individual clusters. Every word is assigned a variable that represents the origin of a global (combination of documents) or local (separate documents) topic. In CluWords, Viegas et al. (2019) utilize nearest words from pre-trained embeddings to create meta-words for document representation and use the Tf-Idf score as weight for weighted clustering. They introduce a novel word representation technique using the syntactic & semantic information obtained from word embedding. Sridhar (2015) present an unsupervised topic model for short texts that employ soft clustering and Gaussian mixture models (GMM) with distributed word representations to overcome sparse word co-occurrence patterns. Another research work (De Miranda et al., 2020) utilizes word2vec to get vector representations of words and implement a mapping mechanism that maps a word vector to a specific topic, aiming to identify topics within texts. However, none of these works utilize standard word embeddings or transformer-based word vectors that are more robust. Sia et al. (2020) apply different clustering processes on various word embedding methods and identify the best combination. But they only consider Tf as corpus statistics.

There are few works (Hasan et al., 2019; Helal and Mouhoub, 2018) in the context of the Bangla language that focuses on just the probabilistic generative approach rather than linguistic attributes. The majority of studies utilize the clustering method for the purpose of developing spell checker (Mandal and Hossain, 2017), identifying syntactically similar words (Ismail and Rahman, 2014), grouping documents based on genre (Ahmad et al., 2018), clustering sentences (Husna et al., 2018), speech recognition (Rahman et al., 2010), etc. In all these studies, clustering in Bangla languages has been performed based on n-gram language models rather than advanced word embedding techniques. Ritu et al. (2018) compare the performances of non-contextualized word embeddings and clustering word vectors, but determining topic words by clustering word vectors is yet to be explored in the context of the Bangla language.

3 Methodology

In this section, we describe our proposed approach and implementation details. Figure 1 depicts the overview of our proposed methodology.

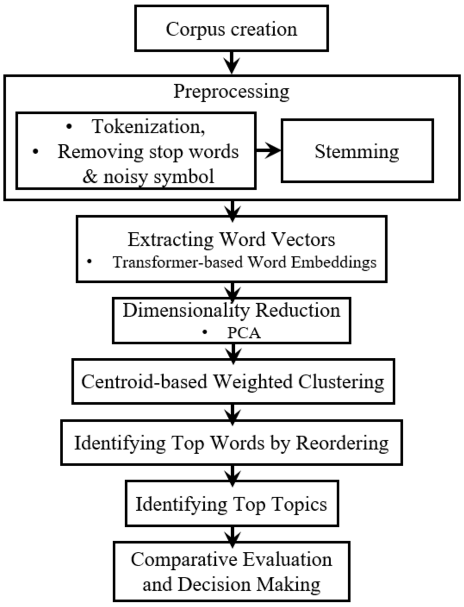

Corpus creation
Preprocessing
• Tokenization,
• Removing stop words & noisy symbol
Stemming
Extracting Word Vectors
• Transformer-based Word Embeddings
Dimensionality Reduction
• PCA
Centroid-based Weighted Clustering
Identifying Top Words by Reordering
Identifying Top Topics
Comparative Evaluation and Decision Making

Figure 1: Workflow diagram of our method

3.1 Corpus Creation

We create our corpus of documents or articles by collecting Bangla news articles from two popular online news portals (e.g., prothom alo[1] and bangla tribune[2]). We collect those news articles using a web crawler that crawls the news text from the HTML pages of respective web pages. For this purpose, we utilize the urllib[3] package and BeautifulSoup[4] library of the Python programming language.

Our corpus contains 197,238 news articles or documents (97,073 documents from “Prothom Alo” and 100,165 from “Bangla Tribune”) dated from 2020 to 2023. Table #1 depicts the category-based distribution of our collected data. Our corpus comprises articles on health, national, international, crime, sports, entertainment, etc. We develop the corpus so that the number of articles in different categories is approximately balanced. Our corpus includes 4,299,788 sentences and 47,856,640 token words from various fields of context. All other statistical overviews of the corpus have been shown in table #2.

3.2 Preprocessing

News articles often contain noisy token sequences and foreign alphabets or symbols. In the preprocessing phase, we tokenize our documents into

[1] https://www.prothomalo.com
[2] https://www.banglatribune.com
[3] https://docs.python.org/3/library/urllib.html
[4] https://pypi.org/project/beautifulsoup4/

Table 1: Category-based distribution of our corpus

Category	Number of Articles	Category	Number of Articles
National	26262	Economics	11767
International	19661	Health	24624
Crime	16785	Religion	15996
Sport	18263	Opinion	13115
Education	16491	Agriculture	15520
Entertainment	18754	**Total**	**197,238**

Table 2: Statistical overview of our corpus

Parameters	Total Amount
Article/Document	197,238
Sentences	4,299,788
Token Words	47,856,640
Unique Token Words	715,670
Average Sentences per Article	21.8
Average Words per Article	242.63
Average Words per Sentence	11.13

tokens, remove stopwords[5], hashtags, IP addresses, URL links, punctuation, and digits. In order to enhance the quality of our analysis, we choose to exclude tokens (i.e., words) that belong to less than five documents and are found inside lengthy sentences exceeding a length of 50 words (Sia et al., 2020). We also eliminate foreign words or symbols, email artifacts, noisy token sequences, etc. Then, we apply stemming to identify the root words of all sub-words and determine the vector representation of root words by averaging the vector representations of their corresponding sub-words. Sometimes, stemming results in a meaningless root word, but it does not affect much as all the sub-words convert to the same root words. After stemming, we find 715,670 unique words (i.e., vocabulary size) in our corpus.

3.3 Word Embedding Methods

In this phase, we extract the vector representations of the vocabulary words from the documents. We choose pre-trained transformer-based Large Language Models (LLM) for extracting those word vectors. The reason behind selecting LLM-based models is their self-attention mechanism. As a result, these models can capture long-term dependencies in textual data. Again, LLM-based models have been pre-trained utilizing large corpora of text containing various contextual information. So, these models can be utilized for any context. At the same time, due to the pre-training with extensive data, LLM-based models can generate standardized representations of word vectors, which makes those models more scalable. In this study, we choose bloom model[6] with 3 billion parameters as the multilingual model and select several BanglaBERT variants (e.g., BanglaBERT (BBert_bha) by (Bhattacharjee et al., 2022), BanglaBERT (BBert_kow) by (Kowsher et al., 2022), and BanglaBERT (BBert_sag) by (Sarker, 2020)) as Bangla LLMs. Another reason behind choosing pre-trained models is that we need not spend time or resource for training models and getting word vectors.

3.4 Dimnesionality Reduction

After converting vocabulary words into word vectors, we get word vectors with a dimension size of 768. This dimension size is identical for all LLM-based models. Due to the sparsity and redundancy of high-dimensional space, clustering algorithms may perform poorly. So, we apply a dimensionality reduction process, Principle Component Analysis (PCA) (Wold et al., 1987), to reduce the dimension size of word vectors. To determine the appropriate dimension size, we examine multiple values ranging from 100 to 700 with an interval of 100.

3.5 Centroid-based Weighted Clustering

We choose to apply centroid-based clustering techniques. Those techniques offer a logical method for obtaining topic words in each cluster by measuring the distance from the cluster center. Again, previous studies suggest that non-centroid-based hierarchical clustering methods lead to inferior performance and necessitate the adjustment of a significant number of hyperparameters (Sia et al., 2020). We apply several clustering techniques

[5] https://github.com/stopwords-iso/stopwords-bn

[6] https://huggingface.co/bigscience/bloom-3b

like KMeans (KM), Spherical k-means (SKM), k-medoids (KMd), and Gaussian Mixture Models (GMM).

We also consider weighted clustering, as the vector representation of words only holds the semantic feature of words but can not understand the corpus statistics (Rahman, 2020b). But word co-occurrence statistics inside a document or corpus are important for identifying topic words. For weighted clustering, we provide weights to our vocabulary words by measuring- Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF).

$$TF(t,d) = \frac{count\ of\ word,\ t\ in\ document,\ d}{total\ number\ of\ words\ in\ document,\ d}$$

$$IDF(t,D) = log(\frac{Number\ of\ documents\ in\ the\ Corpus,\ D}{Number\ of\ documents\ in\ D\ containing\ t+1}+1)$$

$$TF\text{-}IDF(t,d,D) = TF(t,d) * IDF(t,D)$$

3.6 Reordering Clustered Words and Identifying Top X Words

After the clustering phase, we do the reordering of the words in each cluster to identify the top X words. Reordering refers to the organization of the clustered words based on some parameters. These parameters can be any statistical or probabilistic values extracted from the corpus. Clustering based on the data points' similarities & weights, along with the reordering, enhances the appropriateness of the highly relevant topic words inside a topic. Thus, a cluster or topic can be described effectively.

In section 3.5, we get our initial top words of individual clusters by measuring the shortest distance or high similarity between data points and respective cluster centers. We also incorporate the weights of the words during clustering. This procedure does not guarantee to find out the underlying original themes of a cluster or topic (Sia et al., 2020). So, we apply reordering of clustered words based on their average TF & $TF\text{-}IDF$ scores inside particular documents (Sia et al., 2020). Thus, we obtain the top X words from each topic or cluster that can effectively describe the respective topic.

3.7 Getting Top Topics of Documents

Top topics are representative of a particular document, which help understand the context of that document. After getting all the clusters/topics and their corresponding top X words, we determine the top topics of documents or the whole corpus. For identifying the most informative clusters from a particular document, we measure the sum of *euclidean distances* between each cluster center and all word vectors of the document. Finally, we normalize the obtained values for each cluster center using the softmax function to get a probability distribution for all clusters or topics. Clusters with low probabilities obtained from normalization are considered the most informative topics.

4 Experimental Setup

In this section, we discuss the implementation details of our approach.

4.1 Performance Metrics

For measuring the performances of topic modeling methods, we use NPMI (Normalized Point-Wise Mutual Information) (Bouma, 2009) that ranges from [-1, 1] where '1' indicates perfect association, '0' denotes statistical independence, and '-1' represents complete negative association. We only choose NPMI as the performance metric because similar previous studies (Sia et al., 2020) measure only NPMI for measuring performances.

NPMI is a statistical process that measures the direction & strength of association between two words or terms. It is the normalized version of PMI. It measured the extent to which the observed co-occurrence of two terms deviates from what would be expected if the terms were statistically independent. NPMI facilitates a more balanced comparison of associations across different word pairs by providing a bounded range. It is beneficial in topic modeling, where understanding semantic relationships is crucial in extracting meaningful insights from textual data. The formula of NPMI has been shown in the equation 1.

$$NPMI(w_i, w_j) = \frac{log(\frac{P(w_i,w_j)}{(P(w_i)*P(w_j))} + \epsilon)}{-log(P(w_i,w_j) + \epsilon)} \quad (1)$$

$P(w_i, w_j)$ refers to the probability of co-occurrence of words w_i and w_j within the topic and $P(w_i)$ & $P(w_j)$ indicates the probability of the occurring of w_i and w_j within the topic respectively. ϵ is a small smoothing factor.

In our study, we evaluate the NPMI scores of all possible word combinations inside each cluster and average the values to get the average NPMI score of each cluster. This is also called the topic coherence (Blair et al., 2020). Equation 2 depicts the formula of *topic coherence* (Coh) of a topic,

t where N represents the number of topic words inside the topic, t.

$$Coh(t) = \frac{2}{N(N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} NPMI(w_i, w_j) \tag{2}$$

Again, the "average NPMI score for a document" is evaluated by averaging the topic coherence scores of all clusters inside the document.

4.2 Baseline Model and Parameters

We select LDA as our baseline model as it is the most commonly used probabilistic topic modeling technique (Blei et al., 2003). It considers that documents are mixture of latent topics, and each word within a document is assigned to one of these topics. Thus it identifies latent topics, their related word distributions, and the composition of these topics in documents.

To represent a single cluster or topic, we will identify the top 10 topic words. For determining the number of topics, we tune the value of the number of topics, k, from 2 to 20 based on a high average intra-topic similarity score (i.e., topic coherence in equation # 2).

5 Results

In this section, we analyze our experimental results.

5.1 Computational Cost Measurement

We explain the time complexity of our proposed clustering algorithms as well as word embedding methods compared with the probabilistic topic model, LDA.

5.1.1 Time Complexity Analysis for Clustering Algorithms

Table #3 represents the time complexity measurements of different clustering methods. In the table, t is the maximum number of iterations for the worst case, n represents the vocabulary size or the number of data points, k is the number of clusters, and d is the dimension size of the data.

Table 3: Time complexity of clustering algorithms

Algorithms	Initial	Iteration	Overall
KM	$O(kdn)$	$O(tnkd)$	$O(tnkd)$
GMM	-	$O(tnkd^3)$	$O(tnkd^3)$
KMd	$O(kn)$	$O(tkn^2)$	$O(tkn^2)$
SKM	$O(kdn)$	$O(tnkd)$	$O(tnkd)$

t differs based on the clustering algorithms and word embedding methods, as different algorithms require different numbers of iterations for convergence. However, our study considers the t constant factor, which represents the maximum number of iterations for the worst case. Weighted versions of clustering incur an initial cost for weight initialization and introduce a constant factor for remeasuring the cluster centers. The procedure of reordering introduces an additional time complexity of $O(nlog(n_k))$, where n_k represents the average number of words within a cluster.

On the other hand, it is worth noting that the complexity of LDA using collapsed Gibbs sampling is $O(tkN)$, with N representing the total number of tokens inside documents. Consequently, when the value of N far exceeds the value of n, clustering approaches have the potential to offer more favorable trade-offs between performance and complexity.

5.1.2 Word Embeddings Cost

Since we use pre-trained transformer-based word embedding methods, we do not need to train those models from scratch. We need to generate the word vectors for all vocabulary words. So, the tokenized vocabulary words are passed through the transformer layers. The procedure requires linear time complexity. Again, by defining a batch size, this process can be done simultaneously for all batches.

5.2 Best Performed Word Embedding & Clustering Combination

Table #4 presents the average NPMI scores of all the combinations of our proposed transformer-based word embedding techniques and clustering algorithms for our corpus. In this table, the weighted clustering and reordering have been performed based on Tf-Idf scores as those scores improve the performance significantly by statistics ($p = 0.029$; $\alpha = 0.05$ by t-test) rather than Tf scores (Figure 2).

We observe that BBert_kow outperforms other word embedding methods in all conditions as BBert_kow considers 40GB of Bangla textual data during training, which is approximately 1.5 times the corpus size (29.5 GB) used by BBert_bha. Again, the multi-lingual model (Bloom) underperforms due to the low distribution rate of Bangla (0.5%) in their training corpus.

$KM_{w,r}$ shows the best result with all word embeddings among all the clustering variants, as

Table 4: Average NPMI Scores of different combinations of clustering algorithms and word embedding techniques for our corpus

Models		BBert_bha	BBert_kow	BBert_sag	Bloom
Non-weighted Clustering and No reordering	KM	0.07	0.08	0.07	-0.19
	GMM	0.18	0.21	0.19	-0.13
	KMd	0.05	0.05	0.06	-0.22
	SKM	0.06	0.08	0.06	-0.2
Weighted Clustering	KM_w	0.09	0.11	0.09	-0.17
	GMM_w	0.21	0.22	0.22	-0.09
	KMd_w	0.05	0.07	0.06	-0.21
	SKM_w	0.08	0.09	0.08	-0.19
Reordering	KM_r	0.26	0.29	0.28	0.05
	GMM_r	0.25	0.27	0.26	0.01
	KMd_r	0.24	0.26	0.24	-0.02
	SKM_r	0.26	0.27	0.26	0.01
Weighted Clustering and Reordering	$KM_{w,r}$	**0.29**	**0.31**	**0.29**	**0.09**
	$GMM_{w,r}$	0.26	**0.28**	0.28	0.07
	$KMd_{w,r}$	0.25	**0.27**	0.25	0.05
	$SKM_{w,r}$	0.26	**0.29**	0.29	0.07

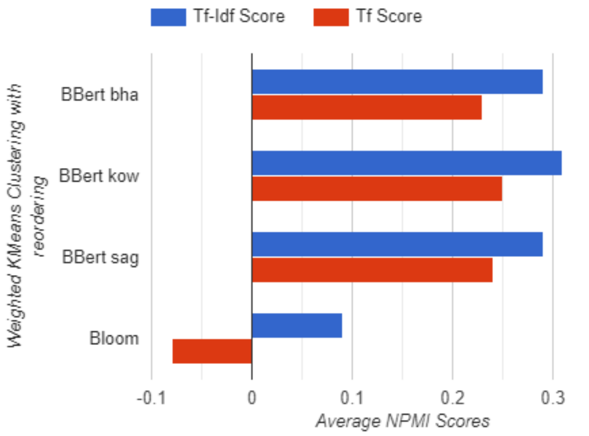


Figure 2: Difference between Tf-Idf and Tf scoring during weighted KMeans clustering and reordering across different word embedding techniques

KMeans perfectly determines cluster centers by averaging clustered word vectors. Weighted clustering and reordering improve the result that is statistically significant ($p = 0.0002$; $\alpha = 0.01$ for $KM_{w,r}$ vs KM). This implies that corpus statistics is an informative attribute for extracting topics. Thus, we determine BBert_kow-$\mathrm{KM_{w,r}}$ as the best-performed combination with the average NPMI score of 0.31.

5.3 Comparison between BBert_kow-$\mathrm{KM_{w,r}}$ and LDA

On the whole corpus, BBert_kow-$\mathrm{KM_{w,r}}$ (average NPMI score= 0.31) also performs significantly better than the traditional topic model (LDA) (average NPMI score= 0.18). We measure the individual NPMI scores of both BBert_kow-$\mathrm{KM_{w,r}}$ and LDA across all documents and find a significant difference with a zero p-value in the t-test between those two topic modeling methods. Figure 3 shows the curves of average NPMI scores for randomly chosen five hundred sample documents using BBert_kow-$\mathrm{KM_{w,r}}$ and LDA methods. In the figure, we observe that the curve for LDA ranges from 0.15 to 0.19, and the curve for BBert_kow-$\mathrm{KM_{w,r}}$ ranges from 0.29 to 0.33.

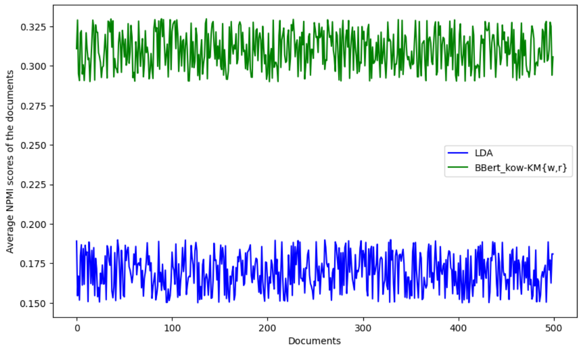


Figure 3: Curves of average NPMI scores for 500 random documents using BBert_kow-$\mathrm{KM_{w,r}}$ and LDA methods

LDA is a generative probabilistic procedure where each topic is a distribution over all vocabulary words. In the topic-word distribution predicted from LDA, the probability of a particular word can be significant for multiple topics. As a result, a common topic word can belong to multiple topics in the LDA method. This scenario will increase the average inter-topic similarity and decrease the average NPMI score of a document.

On the other hand, BBert_kow-$\text{KM}_{\text{w,r}}$ considers hard clustering where a topic word can be included into a cluster or topic. So, all the topic words of a cluster differ from those of the other cluster. For this reason, the average inter-topic similarity is low, and the average NPMI score is high for the BBert_kow-$\text{KM}_{\text{w,r}}$ method. Again, we consider the word embedding of transformer-based LLMs in our proposed method. Transformer-based word embeddings are contextualized word embeddings where a word with different meanings is considered in different contexts. As word embedding holds both the semantic and syntactic characteristics of words, the topic coherence of a cluster increases. As a result, the average NPMI score of the corpus becomes high. Furthermore, corpus statistics are also incorporated during weighted clustering and reordering. Hence, we identify that linguistic attributes (i.e., semantic characteristics of documents) and corpus statistics help extract top-topic words.

The execution time of BBert_kow-$\text{KM}_{\text{w,r}}$ is far less than that of LDA for the whole corpus. For evaluating the entire corpus with 197,343 documents, BBert_kow-$\text{KM}_{\text{w,r}}$ requires 54 seconds, whereas LDA takes 5 minutes 21 seconds on GPU. This empirical study supports the time complexity measurements in Section 5.1 for both our proposed and baseline methods.

5.4 Weighting

Weighted clustering significantly improves the performance of clustering word embeddings. From table #4, we observe the improvement due to the weighting in all cases of clustering and word embedding without reordering. Similarly, the improvement is also visible for all the instances of clustering and word embedding while considering reordering.

Figure 4 presents that the average NPMI score degrades with increased vocabulary size for the entire corpus if weighting is not considered during clustering. With the increase in vocabulary size, words' semantic characteristics are becoming so sparse. This scenario decreases the NPMI score of two words, making identifying words with high topic coherence challenging. Corpus statistics mitigates this issue as weighted clustering. So, weighted clustering based on Tf-Idf is effective for topic analysis from extensive data. The statistical test results for measuring the difference in the performances between weighted clustering and non-weighted clustering are described in appendix A.1.

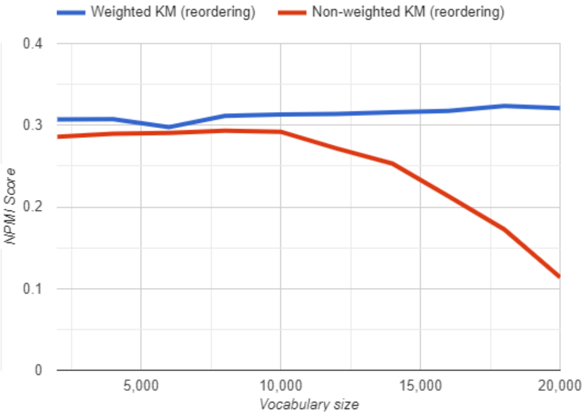


Figure 4: Average NPMI of both weighted and non-weighted clustering with the increase of the vocabulary size for our corpus

5.5 Reordering

Reordering cluster words plays an important role in finding top words per topic. From table #4, we observe that reordering enhances the efficacy of the KMeans clustering more significantly than that of GMM. The statistical test results for measuring the difference in the performances between reordered and non-reordered clustering are described in appendix A.2.

5.6 Performance of PCA

To identify the significance of dimensionality reduction, we consider the weighted KMeans clustering with reordering in different types of word vectors. We examine different dimension sizes that range from 100 to 700 with an interval of 100 and measure the average NPMI score of the whole corpus.

From Figure 5, we find the improvements of the average NPMI scores as the dimension size increases. This is due to the fact that high dimensions can capture more information. However, by observing the elbow points, we can determine that the NPMI values get saturated when the dimension size is 300. So, we consider the dimension size of 300 by applying PCA. Our finding of the dimension size also supports the previous study (Rahman, 2020b) for measuring robust and consistent dimension size of word vectors. In the future, we plan to

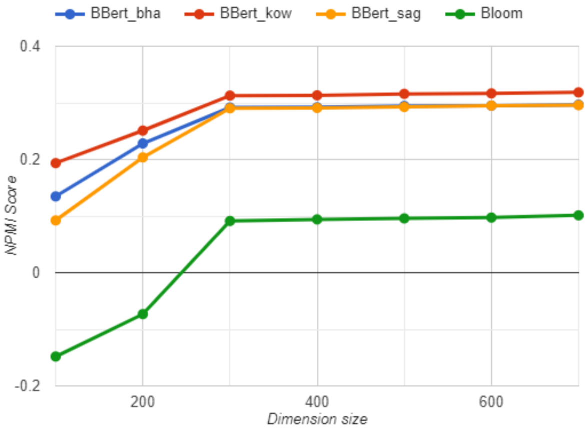


Figure 5: Average NPMI scores of different embedding methods (weighted clustering & reordering) for KMeans with different dimension size for our corpus

apply other neural network-based dimensionality reduction approaches to get better results.

5.7 Qualitative Findings

For the qualitative analysis, we extract the top ten topic words of each topic or cluster from both BBert_kow-KM$_{w,r}$ and LDA. These top topic words describe what the respective topic or cluster actually expresses. We choose top ten topics from both the baseline (i.e., LDA) and our proposed method (i.e., BBert_kow-KM$_{w,r}$). Rather than the real theme exposed by the topics, we determine which genres- the categories that are selected during creating corpus (table #1)- are being covered by those topics. Our findings say that the top ten topics extracted by the BBert_kow-KM$_{w,r}$ technique are different and similar to the ten categories of the corpus. In contrast, LDA retrieve topics from the corpus that represent eight unique genres or categories. The full descriptions of the qualitative results are delineated in appendix B.

6 Limitations

This study has several limitations. First, we use only four language models for word embedding, three of which were Bangla language models. There are also many Bangla LLMs or multi-lingual LLMs with enriched Bangla datasets. These models can be explored in the future. Again, we use pre-trained models to determine word vectors. Integrating dense layers with the LLMs and fine-tuning or pre-training the model for generating word vectors can be a future direction. Second, we only apply centroid-based clustering techniques in our approach. We plan to explore more advanced clustering methods in the future. Another limitation in clustering is the outliers that mislead the position of the cluster centers. A better method to reduce the outliers can positively impact the clustering performance. Third, we only experiment with TF and TF-IDF as corpus statistics. Other informative corpus statistics can be studied in the future. Finally, the neural-network-based dimensionality reduction method can efficiently increase the performance of our proposed topic modeling.

7 Conclusion

Probabilistic generative topic models evaluate word co-occurrences inside documents and ignore linguistic attributes. We apply clustering on word vectors to extract informative topics and compare word embedding techniques and clustering algorithms. By analyzing the whole corpus, the study offers that BanglaBERT (BBert_kow), along with Tf-Idf-based weighted clustering & reordering ($KM_{w,r}$) (average NPMI=0.31 and run-time=54 seconds), outperforms traditional topic models (average NPMI=0.18 and run-time=5 minutes 21 seconds) with respect to top words extraction and time complexity. However, one limitation in clustering is the outliers that mislead the position of cluster centers. We plan to address this issue and perform fine-tuning or pre-training transformer-based word embedding models in the future.

References

Adnan Ahmad, Md Ruhul Amin, and Farida Chowdhury. 2018. Bengali document clustering using word movers distance. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–6. IEEE.

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.

Stuart J Blair, Yaxin Bi, and Maurice D Mulvenna. 2020. Aggregated topic models for increasing social media topic coherence. *Applied Intelligence*, 50:138–156.

David Blei, Lawrence Carin, and David Dunson. 2010. Probabilistic topic models. *IEEE signal processing magazine*, 27(6):55–65.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, 30:31–40.

Jordan Boyd-Graber, Yuening Hu, David Mimno, and 1 others. 2017. Applications of topic models. *Foundations and Trends® in Information Retrieval*, 11(2-3):143–296.

Miriam Cha, Youngjune Gwon, and HT Kung. 2017. Language modeling by clustering with word embeddings for text readability assessment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2003–2006.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804.

Guilherme Raiol De Miranda, Rodrigo Pasti, and Leandro Nunes de Castro. 2020. Detecting topics in documents by clustering word vectors. In *Distributed Computing and Artificial Intelligence, 16th International Conference*, pages 235–243. Springer.

Adji B Dieng, Francisco JR Ruiz, and David M Blei. 2020. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453.

Md Hasan, Md Motaher Hossain, Adnan Ahmed, and Mohammad Shahidur Rahman. 2019. Topic modelling: A comparison of the performance of latent dirichlet allocation and lda2vec model on bangla newspaper. In *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5. IEEE.

MA Helal and Malek Mouhoub. 2018. Topic modelling in bangla language: An lda approach to optimize topics and news classification. *Computer and Information Science*, 11(4):77–83.

Asmaul Husna, Maliha Mostofa, Ayesha Khatun, Jahidul Islam, and Md Mahin. 2018. A framework for word clustering of bangla sentences using higher order n-gram language model. In *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, pages 1–6. IEEE.

Sabir Ismail and M Shahidur Rahman. 2014. Bangla word clustering based on n-gram language model. In *2014 international conference on electrical engineering and information & communication technology*, pages 1–5. IEEE.

Md Kowsher, Abdullah As Sami, Nusrat Jahan Prottasha, Mohammad Shamsul Arefin, Pranab Kumar Dhar, and Takeshi Koshiba. 2022. Bangla-bert: transformer-based efficient model for transfer learning and language understanding. *IEEE Access*, 10:91855–91870.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Prianka Mandal and BM Mainul Hossain. 2017. Clustering-based bangla spell checker. In *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–6. IEEE.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Md Mijanur Rahman, Md Farukuzzaman Khan, and Mohammad Ali Moni. 2010. Speech recognition front-end for segmenting and clustering continuous bangla speech. *Daffodil International University Journal of Science and Technology*, 5(1):67–72.

Rifat Rahman. 2020a. A benchmark study on machine learning methods using several feature extraction techniques for news genre detection from bangla news articles & titles. In *Proceedings of the 7th International Conference on Networking, Systems and Security*, pages 25–35.

Rifat Rahman. 2020b. Robust and consistent estimation of word embedding for bangla language by fine-tuning word2vec model. In *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, pages 1–6. IEEE.

Rifat Rahman, Sheikh Abir Hasan, and Fardous Ahmed Rubel. 2022. Identifying sentiment and recognizing emotion from social media data in bangla language. In *2022 12th International Conference on Electrical and Computer Engineering (ICECE)*, pages 36–39. IEEE.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings. *arXiv preprint arXiv:1906.09821*.

Zakia Sultana Ritu, Nafisa Nowshin, Md Mahadi Hasan Nahid, and Sabir Ismail. 2018. Performance analysis of different word embedding models on bangla language. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5. IEEE.

Sagor Sarker. 2020. Banglabert: Bengali mask language model for bengali language understanding.

Motoki Sato, Austin J Brockmeier, Georgios Kontonatsios, Tingting Mu, John Y Goulermas, Jun'ichi Tsujii, and Sophia Ananiadou. 2017. Distributed document and phrase co-embeddings for descriptive clustering. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 991–1001.

Suzanna Sia, Ayush Dalmia, and Sabrina J Mielke. 2020. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! *arXiv preprint arXiv:2004.14914*.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised topic modeling for short texts using distributed representations of words. In *Proceedings of the 1st workshop on vector space modeling for natural language processing*, pages 192–200.

Felipe Viegas, Sérgio Canuto, Christian Gomes, Washington Luiz, Thierson Rosa, Sabir Ribas, Leonardo Rocha, and Marcos André Gonçalves. 2019. Cluwords: exploiting semantic word clustering representation for enhanced topic modeling. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 753–761.

Chenguang Wang, Mu Li, and Alexander J Smola. 2019. Language models with transformers. *arXiv preprint arXiv:1904.09408*.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Pengtao Xie and Eric P Xing. 2013. Integrating document clustering and topic modeling. *arXiv preprint arXiv:1309.6874*.

He Zhao, Lan Du, and Wray Buntine. 2017. A word embeddings informed focused topic model. In *Asian conference on machine learning*, pages 423–438. PMLR.

A Statistical Test Results for Weighted Clustering and Reordering Cluster Words

A.1 Weighting

Table #5 represents the statistical significant differences between weighted clustering and non-weighted clustering considering all documents. The differences are measured based on the average NPMI scores of documents. We observe significant differences in all the cases when considering reordering. This implies that both the semantic characteristics of words and the corpus statistics play essential roles in determining topic words.

A.2 Reordering

Table #6 depicts the significance of reordering the topic words after weighted clustering. We perform the t-test to measure the difference between reordering and non-reordering. The test is done over all the documents and word embedding methods based on the average NPMI score of documents. In all cases except the Gaussian Mixture Model, reordering causes significant improvement in identifying topic words. This implies that reordering has little impact on weighted GMM as it assumes the mixture of several Gaussian distributions for different clusters where reordering is performed during expectation maximization. Since reordering is performed by default in the GMM, there is no impact of another reordering for extracting topic words.

B Top 10 topics from Corpus & Qualitative discussion

Table #7 and #8 present the topic words for individual topics for BBert_kow-KM$_{w,r}$ and LDA respectively. We find a number of similarities between those two tables while considering the topic words of a particular topic.

From table #7, we identify that the ten topics are mostly similar to the corpus categories. The topic coherence score of the “health” topic is highest. As we collect our corpus for 2020 to 2022 from the archive of the online news portal, the frequency of health-related documents is very high due to the effect of the COVID-19 pandemic. Again, most news articles or documents are relevant to COVID-19. Our findings also reveal that all the topic words under the “Health” topic are related to COVID-19 (table #7). For the same reason, the topic coherence score of this topic is highest. The count of the “National” category news is maximum in our corpus. So, we get the top ten topic words under the “National” topic with a high topic coherence score. The minimum topic coherence score is observed for the “Agriculture” topic. The probable reason behind the low topic coherence score of this topic is the fewer co-occurrences of the topic words inside the agriculture-category documents. The table also displays the topic ranking from top to bottom following the procedure described in the section 3.7. We discover that the “National” topic is the most crucial topic of the corpus. The “National” category is a large domain. We can incorporate new categories (except the “International” category) as the subdomain of the “National” category because all these incidents happen in a nation. So, news of all categories except the “International” category can be considered the “National” category. As a result, the sum of the distances of all the word vectors with respect to the centroid of the “National”

Table 5: t-test result for average NPMI values between non-weighted and weighted clustering considering all individual documents and all word embedding methods

Comparison (without Reordering)	P-value $\alpha = 0.01$	Comparison (with Reordering)	P-value $\alpha = 0.01$
KM Vs KM_w	0.001	KM_r Vs $KM_{w,r}$	0.0001
GMM Vs GMM_w	0.003	GMM_r Vs $GMM_{w,r}$	0.004
KMd Vs KMd_w	0.001	KMd_r Vs $KMd_{w,r}$	0.001
SKM Vs SKM_w	0.002	SKM_r Vs $SKM_{w,r}$	0.001

Table 6: t-test result for average NPMI values between non-reordered and reordered clustering considering all individual documents and all word embedding methods

Comparison (without weighting)	P-value $\alpha = 0.01$	Comparison (with weighting)	P-value $\alpha = 0.01$
KM Vs KM_r	0.0001	KM_w Vs $KM_{w,r}$	0.0003
GMM Vs GMM_r	0.027	GMM_w Vs $GMM_{w,r}$	0.043
KMd Vs KMd_r	0.0005	KMd_w Vs $KMd_{w,r}$	0.0007
SKM Vs SKM_r	0.0003	SKM_w Vs $SKM_{w,r}$	0.0004

Table 7: Topic coherence score of each topic or cluster obtained from BBert_kow-$\mathrm{KM}_{w,r}$

Top 10 topic words for each topic	Topic Coherence score	Topic Name
Politics; Election; Corruption; Hasina; Padma-Bridge; Digital-Bangladesh; Governance; Freedom-Fighters; Awami-League; Dhaka	0.373	National
COVID-19; Vaccine; Lockdown; Delta-Variant; Quarantine; Hospital; Health-Workers; ICU; Mask; Telemedicine	0.385	Health
Rohingya; Myanmar; China; India; UN; Diplomacy; Climate-Change; Trade; SAARC; Refugees	0.334	International
Cinema; OTT Platforms; Dhallywood; Music; Celebrity; Drama; Festival; Television; YouTube; Fashion	0.297	Entertainment
Cricket; BPL; Football; Olympics; Tamim; Shakib; Sports Ministry; Dhaka-League; BCB; World-Cup	0.379	Sport
Rape; Murder; Cybercrime; Trafficking; Corruption; Drug; Robbery; Violence; Police; Arrest	0.318	Crime
Online-Classes; University; HSC; Primary-Education; Scholarship; E-Learning; Reopening; Education-Policy; Ministry-Education; SSC	0.272	Education
GDP; Inflation; Remittance; Export; RMG; Unemployment; Economic-Growth; Budget; SME; Banking	0.257	Economics
Rice; Farmer; Crop; Subsidy; Fisheries; Livestock; Irrigation; Agricultural-Policy; Food-Security; Agrarian-Reform	0.239	Agriculture
Islam; Eid; Mosque; Hindu; Puja; Religious-Freedom; Fatwa; Harmony; Zakat; Madrasa	0.269	Religion

topic or cluster is the lowest, and the "National" topic becomes the most informative topic of our corpus.

Our proposed topic modeling applies KMeans clustering, which is a hard clustering technique. So, the topic words under a topic cannot be in another

Table 8: Topic coherence score of each topic or cluster obtained from LDA

Top 10 topic words for each topic	Topic Coherence score	Topic Name
Advancement; Private; Limited; Institution; Online-classes; MS; Education; Professor; e-Learning; University	0.173	Education
Promise; Human; Sheikh; Bangabadhu; Country; Prime-Minister; Nation; Hasina; Bangladesh; Politics	0.217	National
Secretary; Awami-League; President; Zilla; Sheikh; Chairman; Parliament; Election; Leader; BNP	0.253	National
Corona; Health; Death; Health-Complex; Identification; Sample; Healthy; Hospital; Infection; Treatment	0.238	Health
Police; Arrest; Rescue; Hospital; Union; Corruption; Police-station; OC; Dead-body; Injured	0.158	Crime
Bangladesh; Institution; Dhaka; Bank; Percent; Financial; Limited; Government; Loan; Development	0.147	Economics
Money; Price; Sale; Food; Kilogram; Market; Transaction; Rice; Budget; Onion	0.154	Economics
Cricket; Bangladesh; Test-match; Field; Match; Sport; Captain; Coach; Club; Football	0.177	Sport
August; Complaint; Case; Money; Court; Investigation; Arrest; Lawyer; Mission; Rape	0.162	Crime
India; US; President; China; World; Refugees; International; Organization; Saudi-Arab; UN	0.157	International

topic, and the topics or clusters cannot overlap. This is the main reason for the diversity characteristic of the topics, and we can extract diverse themes from a document or corpus. Another interesting finding (table #7) is that the ten topics determined from our proposed topic modeling are identical to the categories of our corpus except the “Opinion” category. As the news under the “opinion” category are the reflection of various national issues, BBert_kow-KM$_{w,r}$ does not consider this category as a separate cluster.

In table #8, we observe topic names, their topic coherence scores, and the top ten topic words of each topic obtained from the state-of-the-art topic modeling method, LDA. The top two topics with high coherence scores are “Health” and “National”, similar to the finding from table #7. LDA fails to extract diverse topics from the corpus, and we can observe some common words (i.e., money, hospital, etc.) that belong to multiple topics. As LDA predicts the topic-word distribution, a common word can have a high probability value across various topics. It reduces diversity in topics. There are eight unique topics in the table #8 where we extract ten topics from the corpus. The “Entertainment” and “Agriculture” topics are missing when we apply LDA. Some studies (Das et al., 2015) argue that top informative topics from LDA can be identified by their coherence scores. A topic with the highest coherence score is considered the top informative topic. So, the “Health” topic is the most informative topic obtained from LDA. It also supports the result from BBert_kow-KM$_{w,r}$. As the COVID-19 pandemic broke out in early 2020 and continued till 2022, the “Health” topic was highly concerned then.

So, we can decide that our proposed topic modeling method, BBert_kow-KM$_{w,r}$ qualitatively outperforms LDA in understanding the insights of a document or corpus. Again, it can unsupervisedly group the vocabulary words of the corpus into some clusters that are similar to the categories of the corpus.

Benchmarking Large Language Models on Bangla Dialect Translation and Dialectal Sentiment Analysis

Md Mahir Jawad[1], Rafid Ahmed[1], Ishita Sur Apan[2], Tasnimul Hossain Tomal[1], Fabiha Haider[1], Mir Sazzat Hossain[2], Md Farhad Alam Bhuiyan[1]

[1]Penta Global Limited,
[2]Center for Computational & Data Sciences, Independent University, Bangladesh
Correspondence: md.mahir.jawad@g.bracu.ac.bd, ahmedrafid023@gmail.com

Abstract

We present a novel Bangla Dialect Dataset, DialTSA-Bn comprising 600 annotated instances across four major dialects: Chattogram, Barishal, Sylhet, and Noakhali. The dataset was constructed from YouTube comments spanning diverse domains to capture authentic dialectal variations in informal online communication. Each instance includes the original dialectal text, its standard Bangla translation, and sentiment labels (Positive and Negative). We benchmark several state-of-the-art large language models on dialect-to-standard translation and sentiment analysis tasks using zero-shot and few-shot prompting strategies. Our experiments reveal that transliteration significantly improves translation quality for closed-source models, with GPT-4o-mini achieving the highest BLEU score of 0.343 in zero-shot with transliteration. For sentiment analysis, GPT-4o-mini demonstrates near perfect precision, recall, and F1 scores (0.98) in few-shot settings. This dataset addresses the critical gap in resources for low-resource Bangla dialects and provides a foundation for developing dialect-aware NLP systems.

1 Introduction

Bangla, spoken by over 230 million people worldwide, exhibits substantial dialectal variation across different regions of Bangladesh and West Bengal. While standard Bangla has received considerable attention in NLP research, regional dialects remain severely underrepresented in available datasets and models. These dialects differ significantly from standard Bangla in vocabulary, morphology, phonology, and syntax, creating barriers for dialect speakers when interacting with language technologies designed primarily for the standard written form. For instance, the *Vashantor* corpus demonstrates that dialects like Chittagong and Noakhali can diverge strongly in lexical and phonetic space relative to standard Bangla, yielding much lower BLEU scores in dialect to standard translation baselines (Faria et al., 2023).

Beyond structural divergence, dialects also encode deeply rooted region-specific pragmatic and emotional nuances. In different dialects, the same phrase can carry subtly different sentimental intensity or expressive force depending on local idioms, tone, or cultural usage. Such cross-regional variations make sentiment analysis on dialectal text significantly more challenging: a lexical sentiment classifier trained on standard Bangla data may misinterpret or underweight dialect-specific affective markers and discourse cues. Prior work in Bangla sentiment and noisy, informal, social-media text highlights the persistent difficulty of handling dialect drift, slang, and code-mixing (Islam et al., 2021; Alam et al., 2025).

The emergence of large language models (LLMs) has revolutionized natural language processing, yet their performance on low-resource languages and dialects remains inadequately explored. Understanding how modern LLMs handle dialectal variation is crucial for developing inclusive language technologies that serve diverse linguistic communities. Furthermore, the lack of high-quality annotated datasets for Bangla dialects has impeded progress in this domain. A particularly intriguing and underexplored aspect is how the script itself influences model performance: initial observations suggest that even powerful models may struggle with the morphological complexities of non-Latin scripts like Bangla, but may unlock superior capabilities when the input is transliterated into a familiar Latin representation. For Bangla specifically, the *BanglaTLit* benchmark demonstrates that back-transliteration techniques can help align Romanized and native-script forms in downstream tasks (Fahim et al., 2024).

This paper addresses these challenges by introducing a new Bangla Dialect Dataset collected from authentic online communication on YouTube.

Our dataset covers four major dialects: Chattogram, Barishal, Sylhet, and Noakhali, each with 150 annotated instances. We contribute both dialectal texts and their standard Bangla translations, along with sentiment annotations, creating a multi-task resource that links dialectal variation with both translation and sentiment.

We conduct comprehensive experiments evaluating multiple state-of-the-art LLMs on two tasks: dialect-to-standard translation and sentiment analysis. Our investigation includes both closed-source models (Gemini 2.5 Flash (Comanici et al., 2025), GPT-4o-mini (OpenAI, 2024), Claude (Anthropic, 2024)) and open-source models (Qwen-2.5-7B (Qwen Team, 2024), Gemma-3-12B (Gemma Team, 2024), Llama-3.1-8B (Meta AI, 2024), Mistral (Jiang et al., 2023)) under zero-shot and few-shot conditions. A key part of our analysis examines the impact of transliteration on performance, revealing critical insights about how script representation affects the processing of non-Latin text. In particular, we observe that closed-source models in many cases see dramatic performance gains when dialectal input is presented in Latin script.

Our main contributions are: (1) a novel annotated dataset of 600 instances covering four major Bangla dialects with translations and sentiment labels, (2) comprehensive benchmarking of modern LLMs on dialectal Bangla tasks, (3) empirical evidence demonstrating the significant impact of transliteration on translation quality, highlighting a potential bottleneck in cross-lingual transfer for non-Latin scripts, and (4) insights into how dialectal variation intersects with sentiment interpretation, pointing toward dialect-aware NLP systems for low-resource languages.

2 Related Work

Research on Bangla dialect processing has accelerated only recently, with several resources tackling dialect→standard conversion and regional variation. The *Vashantor* benchmark covers multiple Bangla regional dialects and provides parallel dialect→standard data (Faria et al., 2023). Closer to specific regions, *ChatgaiyyaAlap* releases Chittagonian↔Standard Bangla pairs suitable for normalization and translation (Chowdhury et al., 2025), while *ONUBAD* broadens coverage with datasets from Chittagong, Sylhet, and Barishal including glosses (Sultana et al., 2025). Beyond MT/normalization, *ANCHOLIK-NER* introduces a regional NER benchmark for Bangla, indicating growing interest in dialect-aware evaluation (Paul et al., 2025a,b). Related spoken-language understanding data also captures colloquial Bangla and Sylheti for intent/slot modeling (Sakib et al., 2023).

Dialect-to-standard normalization connects to broader multilingual work that treats normalization as distinct from generic text cleaning. In Arabic, large-scale efforts such as *MADAR* assemble 25-city dialect corpora aligned with MSA (Bouamor et al., 2018), and community evaluations have benchmarked dialect identification and dialect→standard transfer (Elneima et al., 2024; Abdul-Mageed et al., 2021). These lines of work establish methodological precedents for evaluation protocols and reporting.

Script choice has emerged as a key factor. Studies show that transliteration to a familiar Latin representation can substantially improve performance for models trained predominantly on Latin-script data. Systematic investigations of in-context learning report consistent gains from transliteration for low-resource, non-Latin scripts (Ma et al., 2024); for Bangla specifically, *BanglaTLit* offers a benchmark for back-transliteration of Romanized Bangla, enabling controlled analysis of script effects and error propagation (Fahim et al., 2024). More broadly, context-aware transliteration methods for Romanized South Asian languages demonstrate sentence-level modeling relevant to noisy user-generated text (Kirov et al., 2024). Recent work also proposes reversible, compression-friendly transliteration frameworks that can facilitate cross-lingual transfer at scale (Zhuang et al., 2025).

For sentiment analysis on informal Bangla text, prior datasets highlight the challenges of noise, code-mixing, and dialectal drift. *SentNoB* compiles noisy social-media comments with three-way sentiment labels (Islam et al., 2021), while *BnSentMix* focuses on Bengali–English code-mixed sentiment (Alam et al., 2025).

Taken together, existing work establishes the importance of dialect-aware resources, normalization to standard varieties, and the non-trivial impact of script choice. Our contributions complement this landscape by releasing a focused, multi-dialect Bangla dataset with aligned translations and sentiment labels, and by providing a controlled analysis of transliteration effects on modern LLMs across translation and sentiment tasks.

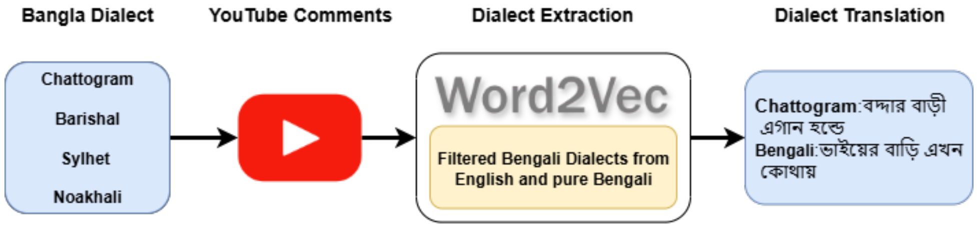

Bangla Dialect
YouTube Comments
Dialect Extraction
Dialect Translation
Chattogram
Barishal
Sylhet
Noakhali
Word2Vec
Filtered Bengali Dialects from English and pure Bengali
Chattogram:বদ্দার বাড়ী এগান হন্ডে
Bengali:ভাইয়ের বাড়ি এখন কোথায়

Figure 1: Dataset generation pipeline showing YouTube comment filtering and manual translation of four Bangla dialects into standard Bangla.

3 Bangla Dialect: Dataset Creation

We constructed the DIALTSA-BN dataset covering four major Bangla dialects: **Chattogram**, **Barishal**, **Sylhet**, and **Noakhali**. Each dialect set contains **150 instances**, resulting in a total of **600 annotated samples**. Each instance includes a dialectal text, its standard Bangla translation, and a sentiment label.

3.1 Data Collection

The data was collected from *YouTube comments* across diverse domains such as movies, dramas, vlogs, news, debates, and music videos. Using the YouTube API, we crafted region-specific queries to capture naturally occurring dialectal variations in informal online communication. This ensured broad coverage of socio-cultural and linguistic diversity across regions.

3.2 Data Preprocessing

Collected comments were cleaned by removing URLs, emojis, and other non-textual artifacts. To distinguish dialectal Bangla from standard Bangla and code-mixed text, we employed a *Word2Vec embedding model*. A similarity threshold of **70%** was applied, filtering out overly standard sentences while preserving authentic dialectal patterns. This threshold ensures removing standard Bangla comments and only keep Bangla words that belongs to certain dialects and not match up with standard Bangla. The resulting dataset emphasizes phonological and lexical diversity across regions.

3.3 Data Annotation

Annotation was conducted in two stages. First, each dialectal sentence was manually translated into **standard Bangla** to enable downstream translation evaluation. The translation was done by individual experts of specific dialects. Four different persons with efficiency in Sylhet, Noakhali, Chattogram and Barishal dialect as well as fluent standard Bangla speakers were appointed to fulfill the annotation. Second, annotators assigned one of three sentiment labels, *Positive*, *Negative*, or *Neutral*, to each instance. The final dataset thus provides parallel dialect–standard pairs and sentiment annotations, supporting both translation and sentiment analysis research on dialectal Bangla.

4 Methodology

Our evaluation methodology encompasses two primary tasks: dialect-to-standard Bangla translation and sentiment analysis. We systematically assess model performance under different prompting strategies and input representations, with all prompt templates provided in the Appendix for full transparency.

4.1 Prompting Strategies

Zero-Shot Prompting. In this baseline setup, models receive only task-specific instructions and the input dialectal text without any example demonstrations. For translation, the prompt instructs the model to translate the dialect text into standard Bangla while preserving meaning and tone. For sentiment classification, the prompt requests a binary prediction of *POSITIVE* or *NEGATIVE*. This setup evaluates a model's intrinsic ability to generalize from pretraining to unseen dialectal data.

Few-Shot Prompting. To provide contextual cues, we include example input–output pairs before the target query. For each test instance, five examples from each of the four dialects (20 total) are randomly sampled from the dataset. Each example consists of a dialectal sentence, its standard Bangla translation, and a sentiment label. In few-shot translation and sentiment classification, these examples are embedded within the prompt, allowing models to better capture dialectal patterns and sentiment cues with minimal supervision.

4.2 Transliteration Experiments

To examine how script representation affects model performance, we conduct parallel experiments with and without transliteration. In the transliterated condition, dialectal Bangla sentences are manually converted into Latin script by trained annotators while preserving phonetic and regional characteristics. Both the native Bangla text and its transliteration are provided in the prompt, allowing the model to leverage cross-script alignment. This design tests the hypothesis that models pretrained predominantly on Latin-script corpora may better interpret transliterated Bangla inputs, improving both translation fluency and sentiment consistency.

4.3 Evaluation Metrics

Translation Quality. We use five complementary metrics to evaluate translation: BLEU, ROUGE-1, ROUGE-2, ROUGE-L, and METEOR. BLEU measures n-gram precision, while ROUGE captures recall-based overlap of lexical units and sequences. METEOR incorporates synonym matching and word-order flexibility, offering a more semantic measure of translation accuracy. Together, these metrics provide a balanced view of lexical and semantic fidelity.

Sentiment Analysis. For sentiment evaluation, we report precision (P), recall (R), and F1-score (F1) across all sentiment classes. Model predictions are compared against human-annotated gold labels. All metrics are averaged across the four dialects, Chattogram, Barishal, Sylhet, and Noakhali, to ensure balanced performance assessment without regional bias.

5 Experimental Setup

We benchmark seven large language models on the DIALTSA-BN dataset: three closed-source models Gemini 2.5 Flash (Comanici et al., 2025), GPT-4o-mini (OpenAI, 2024), and Claude (Anthropic, 2024) and four open-source models Qwen-2.5-7B (Qwen Team, 2024), Gemma-3-12B (Gemma Team, 2024), Llama-3.1-8B (Meta AI, 2024), and Mistral (Jiang et al., 2023). All experiments are conducted on the full dataset of 600 annotated samples covering four Bangla dialects: Chattogram, Barishal, Sylhet, and Noakhali.

Closed-source models are accessed through the OpenRouter API, while open-source models are deployed on Lightning AI cloud GPUs. To ensure consistency and reproducibility, inference parameters are standardized across all runs, with the temperature fixed at 0.1 to minimize randomness and the maximum token length set to 64 to accommodate complete translations and sentiment outputs.

Each model is evaluated under four configurations: (1) zero-shot prompting, (2) few-shot prompting, (3) zero-shot prompting with transliteration, and (4) few-shot prompting with transliteration. Results are averaged across the four dialects to provide aggregate performance metrics representing overall cross-dialect generalization.

6 Result and Analysis

Our experiments reveal significant performance variations across models, prompting strategies, and input representations. We present key findings organized by task and experimental condition.

6.1 Translation Performance

Zero-Shot Results. From Table 1, we observe that translation quality remains low across all models in the zero-shot setting, indicating the difficulty of dialect-to-standard Bangla translation without prior exposure. Among closed-source models, Claude performs best (BLEU = 0.064, ROUGE-L = 0.686), followed by GPT-4o-mini (BLEU = 0.023, ROUGE-L = 0.411), while Gemini 2.5 Flash shows the weakest performance.

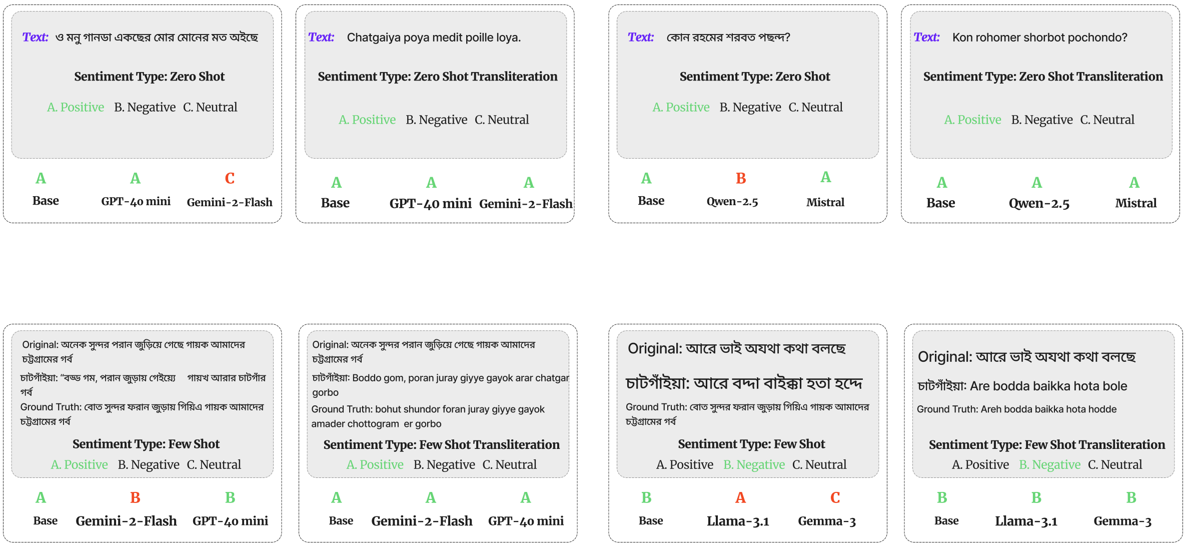
Among open-source models, Llama-3.1-8B (BLEU = 0.083, ROUGE-L = 0.560) and Mistral (BLEU = 0.069, ROUGE-L = 0.548) achieve comparable results, with Mistral also attaining the highest METEOR score (0.245). These findings suggest that recent open-source models can perform on par with, or slightly better than, closed-source ones in zero-shot dialect translation.

Few-Shot Results. As shown in Table 1, few-shot prompting substantially improves translation performance across all models. Among closed-source LLMs, Claude achieves the highest BLEU score (0.046) and ROUGE-L (0.525), followed closely by GPT-4o-mini (BLEU = 0.039, ROUGE-L = 0.507) and Gemini 2.5 Flash (BLEU = 0.051, ROUGE-L = 0.451). These results indicate that providing examples enables better handling of dialectal variations compared to the zero-shot setting.

For open-source models, the improvements are more pronounced. Llama-3.1-8B (BLEU = 0.112, ROUGE-L = 0.653, METEOR = 0.297) and Mistral (BLEU = 0.109, ROUGE-L = 0.689, METEOR = 0.325) outperform all closed-source coun-

Models	**Translation**					**Sentiment**		
	BLEU	**R1**	**R2**	**R_L**	**Meteor**	**P**	**R**	**F1**
Zero Shot Prompt								
Closed Source VLMs								
Gemini 2.5 Flash	0.010	0.201	0.151	0.185	0.087	0.594	0.443	0.491
GPT-4o-mini	0.023	0.448	0.311	0.411	0.145	0.596	0.501	0.530
Claude	0.064	0.727	0.531	0.686	0.188	0.581	0.375	0.424
Open Source VLMs								
Qwen-2.5-7B	0.032	0.617	0.285	0.541	0.063	0.628	0.608	0.554
Gemma-3-12B	0.048	0.669	0.413	0.603	0.122	0.537	0.547	0.534
Llama-3.1-8B	0.083	0.618	0.391	0.560	0.217	0.460	0.474	0.292
Mistral	0.069	0.579	0.426	0.548	0.245	0.595	0.591	0.605
Zero Shot with Transliteration								
Closed Source VLMs								
Gemini 2.5 Flash	0.215	0.545	0.470	0.532	0.425	0.599	0.452	0.504
GPT-4o-mini	0.317	0.886	0.810	0.878	0.582	0.626	0.566	0.587
Claude	0.330	0.843	0.761	0.829	0.571	0.535	0.385	0.420
Open Source VLMs								
Qwen-2.5-7B	0.035	0.634	0.293	0.556	0.065	0.628	0.608	0.554
Gemma-3-12B	0.048	0.669	0.413	0.603	0.122	0.537	0.547	0.534
Llama-3.1-8B	0.080	0.595	0.376	0.539	0.209	0.460	0.474	0.292
Mistral	0.072	0.608	0.447	0.575	0.257	0.595	0.591	0.605
Few Shot Prompt								
Closed Source VLMs								
Gemini 2.5 Flash	0.051	0.486	0.370	0.451	0.212	0.953	0.942	0.947
GPT-4o-mini	0.039	0.541	0.388	0.507	0.175	0.98	0.98	0.98
Claude	0.046	0.561	0.406	0.525	0.163	0.764	0.747	0.754
Open Source VLMs								
Qwen-2.5-7B	0.032	0.614	0.282	0.540	0.061	0.656	0.645	0.639
Gemma-3-12B	0.083	0.725	0.530	0.679	0.229	0.682	0.709	0.639
Llama-3.1-8B	0.112	0.702	0.501	0.653	0.297	0.624	0.650	0.538
Mistral	0.109	0.719	0.567	0.689	0.325	0.494	0.498	0.479
Few Shot with Transliteration								
Closed Source VLMs								
Gemini 2.5 Flash	0.063	0.606	0.476	0.569	0.222	0.793	0.723	0.752
GPT-4o-mini	0.068	0.625	0.459	0.598	0.198	0.98	0.98	0.98
Claude	0.066	0.639	0.471	0.601	0.188	0.647	0.622	0.630
Open Source VLMs								
Qwen-2.5-7B	0.032	0.614	0.282	0.540	0.061	0.656	0.645	0.639
Gemma-3-12B	0.059	0.734	0.488	0.677	0.142	0.629	0.633	0.614
Llama-3.1-8B	0.108	0.675	0.482	0.628	0.286	0.624	0.650	0.538
Mistral	0.114	0.755	0.595	0.723	0.341	0.494	0.498	0.479

Table 1: Benchmarking of LLMs on the DialTSA-Bn dataset across translation and sentiment tasks under zero-shot and few-shot prompting, with and without transliteration. Scores are averaged over four major Bangla dialects: Chattogram, Barishal, Sylhet, and Noakhali.



Text: ও মনু গানডা একছের মোর মোনের মত অইছে

Sentiment Type: Zero Shot

A. Positive B. Negative C. Neutral

Text: Chatgaiya poya medit poille loya.

Sentiment Type: Zero Shot Transliteration

A. Positive B. Negative C. Neutral

Text: কোন রহমের শরবত পছন্দ?

Sentiment Type: Zero Shot

A. Positive B. Negative C. Neutral

Text: Kon rohomer shorbot pochondo?

Sentiment Type: Zero Shot Transliteration

A. Positive B. Negative C. Neutral

Original: অনেক সুন্দর পরান জুড়িয়ে গেছে গায়ক আমাদের চট্টগ্রামের গর্ব

চাটগাঁইয়া: "বড্ড গম, পরান জুড়ায় গেইয়্যে গায়খ আরার চাটগাঁর গর্ব

Ground Truth: বোত সুন্দর ফরান জুড়ায় গিয়িএ গায়ক আমাদের চট্টগ্রামের গর্ব

Sentiment Type: Few Shot

A. Positive B. Negative C. Neutral

Original: অনেক সুন্দর পরান জুড়িয়ে গেছে গায়ক আমাদের চট্টগ্রামের গর্ব

চাটগাঁইয়া: Boddo gom, poran juray giyye gayok arar chatgar gorbo

Ground Truth: bohut shundor foran juray giyye gayok amader chottogram er gorbo

Sentiment Type: Few Shot Transliteration

A. Positive B. Negative C. Neutral

Original: আরে ভাই অযথা কথা বলছে

চাটগাঁইয়া: আরে বদ্দা বাইক্কা হতা হদ্দে

Ground Truth: বোত সুন্দর ফরান জুড়ায় গিয়িএ গায়ক আমাদের চট্টগ্রামের গর্ব

Sentiment Type: Few Shot

A. Positive B. Negative C. Neutral

Original: আরে ভাই অযথা কথা বলছে

চাটগাঁইয়া: Are bodda baikka hota bole

Ground Truth: Areh bodda baikka hota hodde

Sentiment Type: Few Shot Transliteration

A. Positive B. Negative C. Neutral

Figure 2: Error Analysis of open and closed source models

terparts, showing stronger contextual learning abilities. Gemma-3-12B also demonstrates solid performance (BLEU = 0.083, ROUGE-L = 0.679), while Qwen-2.5-7B remains comparatively weaker. Overall, few-shot prompting enhances lexical and semantic alignment, narrowing the performance gap between open- and closed-source models.

Impact of Transliteration. Adding transliteration to the input text significantly improves translation quality in both zero-shot and few-shot settings. In the zero-shot setup, BLEU and ROUGE scores rise sharply, from below 0.10 to over 0.30 for top-performing models, showing that transliteration helps models better interpret dialectal words by aligning them with familiar phonetic patterns.

In the few-shot setting, the gains are smaller but consistent. Models show notable improvements in BLEU and METEOR, reflecting better lexical and semantic alignment. Overall, transliteration within the prompt acts as a simple yet effective cue that enhances the model's grasp of dialectal phonetics, resulting in more accurate and fluent translations.

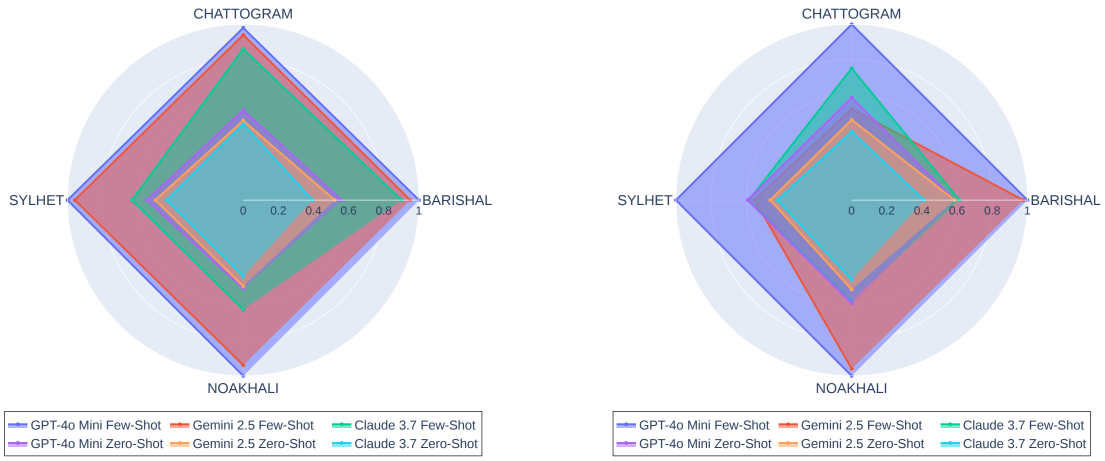
Error Analysis An error analysis was conducted to compare the performance of various Large Language Models (LLMs) against ground-truth sentiment labels for Bangla dialects. As illustrated in **Figure 2** of the study, this qualitative analysis spanned multiple conditions, including zero-shot and few-shot prompting, both with and without *transliteration*. The analysis revealed significant performance variations, particularly highlighting that dialects with greater phonetic divergence from standard Bangla, such as **Chattogram** and **Barishal**, were more challenging for the models and resulted in lower scores. The figure provides concrete examples of misclassifications, such as models incorrectly identifying sentiment polarity (e.g., predicting 'Positive' when the ground truth was 'Negative'), and demonstrates instances where prompting strategies or transliteration helped to correct these errors.

6.2 Sentiment Analysis Performance

Zero-Shot Results. As shown in Table 1, zero-shot sentiment performance remains moderate across all models. Among closed-source models, GPT-4o-mini achieves the best F1 score (0.530), followed by Gemini 2.5 Flash (0.491) and Claude (0.424). For open-source models, Mistral performs the highest (0.605), with Qwen-2.5-7B close behind (0.554). These results indicate that models can capture general sentiment polarity but often misclassify subtle or context-dependent emotions without examples.

Few-Shot Results. Few-shot prompting leads to a large performance gain across all models. GPT-4o-mini achieves perfect accuracy (P = 0.98, R = 0.98, F1 = 0.98), followed by Gemini 2.5 Flash (F1 = 0.947) and Claude (F1 = 0.754). Among open-source models, Gemma-3-12B and Qwen-2.5-7B reach F1 scores around 0.64, outperforming Llama-3.1-8B and Mistral. This demonstrates that in-context examples enhance the models' ability to associate emotional cues with textual context.

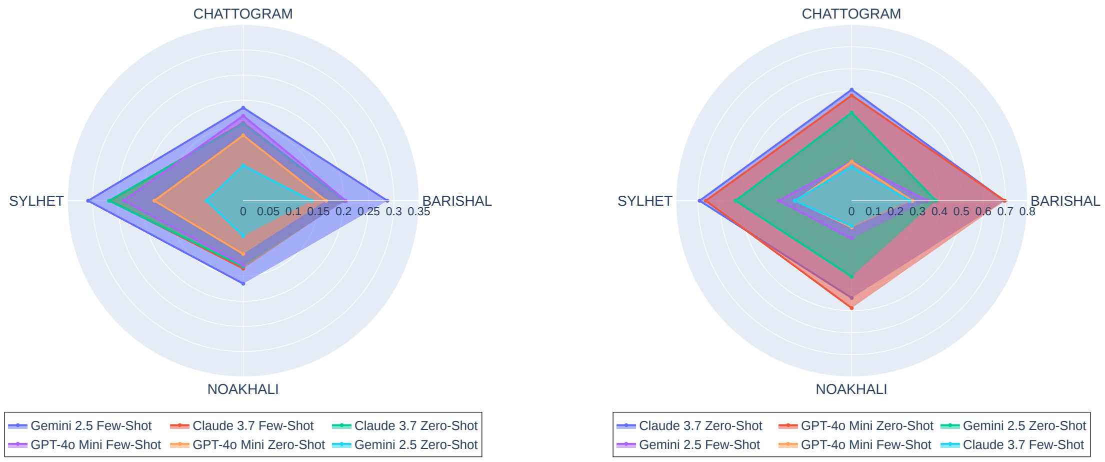
Impact of Transliteration. Adding translitera-



(a) F1 score for base dataset

(b) F1 score for transliterated dataset

Figure 3: F1 score comparison of base dataset and transliterated dataset for Zero-Shot and Few-Shot prompts across different closed source models



(a) Meteor score for base dataset

(b) Meteor score for transliterated dataset

Figure 4: Meteor comparison of base dataset and transliterated dataset for Zero-Shot and Few-Shot prompts across different closed source models

tion yields small but consistent improvements in the zero-shot setting, with GPT-4o-mini improving to an F1 of 0.587. However, in the few-shot setup, the effect is marginal, as models already perform strongly with example-based prompting. Overall, transliteration helps slightly refine sentiment recognition in zero-shot scenarios but offers limited benefit when contextual examples provided.

7 Discussion

7.1 Script Representation Matters

A key finding of this study is that script representation strongly affects model performance. Adding transliteration within the prompt consistently improves translation quality and, to a lesser extent, sentiment classification. Transliteration converts dialectal Bangla into a standardized phonetic form that aligns better with models' subword vocabularies, reducing tokenization errors and improving lexical matching.

These results highlight how text is represented, whether in native Bangla script or Romanized transliteration, directly impacts model comprehension and output quality. The improved performance with transliterated input suggests that most large models have stronger familiarity with Latin-script tokens, enabling better cross-dialect alignment and semantic interpretation.

7.2 Task Complexity Differences

Although both tasks involve dialectical comprehension, their linguistic demands differ substantially. Translation requires accurate lexical and syntactic alignment between dialectal and standard Bangla, making it more sensitive to orthographic and morphological variations. Minor phonetic differences can lead to significant semantic deviations, explaining the lower BLEU and ROUGE scores observed in zero-shot settings.

In contrast, sentiment classification depends more on overall tone and contextual cues than exact word forms. Models can often infer polarity even from partially understood text, which explains their relatively stable performance. Translation thus represents a formally constrained and representation-dependent task, while sentiment classification benefits from contextual reasoning and in-context cues.

7.3 Dialectal Variation Effects.

The radar plots in Figures 3 and 4 illustrate model performance across the four major dialect regions, Chattogram, Barishal, Sylhet, and Noakhali. Performance varies notably by region, reflecting differences in lexical forms, phonetic structures, and orthographic patterns among dialects. Models tend to perform better on Barishal and Sylhet, which share greater lexical similarity with standard Bangla, while Chattogram and Noakhali exhibit lower scores due to stronger phonetic divergence.

These results highlight that dialectal variation remains a key source of difficulty for both translation and sentiment analysis tasks. Models trained primarily on standard Bangla struggle to interpret dialect-specific tokens or informal constructions, leading to inconsistent performance across regions. Addressing this gap will require dialect-aware datasets and modeling approaches that explicitly capture regional linguistic diversity.

7.4 Role of In-Context Learning.

Few-shot prompting emerges as another key factor in improving performance. For translation, BLEU and ROUGE scores rise notably when sample pairs are provided, showing stronger word alignment and contextual comprehension. In sentiment classification, F1 scores increase sharply, reaching perfect accuracy for GPT-4o-mini and strong performance for Gemini 2.5 Flash, indicating that even minimal contextual exposure enables better sentiment recognition in dialectal text. Together with transliteration, in-context learning demonstrates that strategic prompting can significantly enhance model robustness without task-specific fine-tuning.

8 Conclusion

We have presented a novel DIALTSA-BN comprising 600 carefully annotated instances across four major dialects: Chattogram, Barishal, Sylhet, and Noakhali, collected from authentic and diverse YouTube comments. This dataset fills a critical gap in resources for low-resource Bangla dialectal NLP by providing dialectal texts, standard Bangla translations, and sentiment annotations. Through comprehensive benchmarking of seven state-of-the-art large language models, we uncover several key insights into dialectal language understanding and representation. Our experiments show that **transliteration** markedly enhances translation quality for closed-source models, with GPT-4o-mini achieving a BLEU score of 0.343 when dialectal input is rendered in Latin script. **Few-shot prompting** proves highly effective for sentiment analysis, yielding perfect F1-scores for GPT-4o-mini, while translation remains challenging even with contextual examples. These findings highlight both the potential and the limitations of current LLMs, indicating that they can successfully capture dialectal sentiment but still struggle with high-fidelity dialect-to-standard translation. The DIALTSA-BN dataset and benchmarks lay a strong foundation for developing inclusive, dialect-aware, and culturally adaptive Bangla language technologies, helping bridge the divide between standard and regional language users.

Limitations and Future Work

This work has several limitations. The dataset size of 600 instances, while carefully curated and manually annotated, remains relatively small for training robust or generalized models. As a low-resource language with limited digital presence, Bangla dialects pose inherent challenges for large-scale data collection. Moreover, YouTube comments, though authentic and diverse, may not fully capture the phonetic richness and conversational variety found in real spoken interactions. Consequently, some dialectal expressions and regional subtleties are underrepresented in the current dataset.

Future work should expand DIALTSA-BN to include additional dialects beyond the four studied here, such as those spoken in Mymensingh, Rangpur, and Khulna, to achieve more comprehensive dialectal coverage. Enhancing translation quality through linguistically informed preprocessing or dialect-specific normalization remains an important direction. Fine-tuning or training compact, domain-adapted models on dialectal data could further improve efficiency and accessibility. In addition, exploring multi-task or transfer learning frameworks that jointly optimize translation and sentiment tasks may yield better cross-dialect generalization. Finally, extending this framework to other low-resource languages with similar dialectal diversity would strengthen the generalizability and broader impact of this research.

References

Muhammad Abdul-Mageed, Chiyu Zhang, AbdelRahim Elmadany, Houda Bouamor, and Nizar Habash. 2021. NADI 2021: The second nuanced arabic dialect identification shared task. In *Proceedings of WANLP 2021*.

Sadia Alam, Md Farhan Ishmam, Navid Hasin Alvee, Md Shahnewaz Siddique, Md Azam Hossain, and Abu Raihan Mostofa Kamal. 2025. BnSentMix: A diverse bengali–english code-mixed dataset for sentiment analysis. In *Proceedings of the 8th Workshop on Low-Resource Language Processing (LoResLM 2025)*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. Technical report, Anthropic.

Houda Bouamor, Nizar Habash, Mohammad Salameh, et al. 2018. The MADAR arabic dialect corpus and lexicon. In *Proceedings of LREC 2018*. 25-city dialects + MSA parallel resources.

S. Chowdhury, M. Rahman, et al. 2025. Chatgaiyyaalap: A dataset for conversion from chittagonian dialect to standard bangla. *Data in Brief*. Dataset and paper describing 4,012 SB–Chittagonian pairs.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Abdulrahman H. Elneima et al. 2024. Osact6 dialect to MSA translation shared task overview. In *Proceedings of the Sixth Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT6), EACL 2024*.

Md Fahim, Farhan Ishmam Islam, et al. 2024. BanglaTLit: A benchmark dataset for back-transliteration of romanized bangla. In *Findings of EMNLP 2024*.

F.T.J. Faria, M. Mukaffi, M. Rahman, et al. 2023. Vashantor: A large-scale multilingual benchmark dataset for bangla regional dialects. *arXiv preprint arXiv:2311.11142*.

Gemma Team. 2024. Gemma: Open models based on gemini research and technology. Technical Report, Google DeepMind.

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.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Christo Kirov, Cibu Johny, Anna Katanova, Alexander Gutkin, and Brian Roark. 2024. Context-aware transliteration of romanized south asian languages. *Computational Linguistics*, 50(2):475–534.

Chunlan Ma, Yihong Liu, Haotian Ye, and Hinrich Schütze. 2024. Exploring the role of transliteration in in-context learning for low-resource languages written in non-latin scripts. *arXiv preprint arXiv:2407.02320*.

Meta AI. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Bidyarthi Paul, Faika Fairuj Preotee, Shuvashis Sarker, Shamim Rahim Refat, Shifat Islam, Tashreef Muhammad, Mohammad Ashraful Hoque, and Shahriar Manzoor. 2025a. Ancholik-ner: A benchmark dataset for bangla regional named entity recognition. *arXiv preprint arXiv:2502.11198*.

Bidyarthi Paul et al. 2025b. Ancholik-ner (dataset release). Mendeley Data.

Qwen Team. 2024. Qwen 2.5: A party of foundation models. Technical Report, Alibaba Cloud. Available at `https://qwenlm.github.io/blog/qwen2.5/`.

F.A. Sakib, M.R. Karim, M.K. Hasan, et al. 2023. Intent detection and slot filling for home assistants: Formal bangla, colloquial bangla and sylheti. *arXiv preprint arXiv:2310.10935*.

N. Sultana, S. Akter, et al. 2025. ONUBAD: A comprehensive dataset for automated conversion of bangla dialects to standard bangla. *Data in Brief*. Chittagong, Sylhet, Barishal; words/clauses/sentences with English.

Wenhao Zhuang, Yuan Sun, and Xiaobing Zhao. 2025. Enhancing cross-lingual transfer through reversible transliteration: A huffman-based approach for low-resource languages. In *Proceedings of ACL 2025*. Reversible Latin transliteration framework; efficiency + accuracy.

A Further Experiment Results

This appendix presents comprehensive performance metrics and prompting strategies for each dialect across different models and experimental conditions.

Dialect	Model	Acc	Prec	Rec	F1
	Zero-Shot				
Barishal	Gemini 2.5 Flash	0.740	0.635	0.454	0.520
	Claude 3.7 Sonnet	0.560	0.619	0.317	0.394
	GPT-4o-mini	0.860	0.615	0.524	0.559
Chattogram	Gemini 2.5 Flash	0.680	0.499	0.441	0.452
	Claude 3.7 Sonnet	0.620	0.504	0.404	0.432
	GPT-4o-mini	0.760	0.549	0.497	0.511
Sylhet	Gemini 2.5 Flash	0.640	0.600	0.453	0.500
	Claude 3.7 Sonnet	0.540	0.594	0.413	0.436
	GPT-4o-mini	0.760	0.597	0.538	0.548
Noakhali	Gemini 2.5 Flash	0.660	0.641	0.422	0.490
	Claude 3.7 Sonnet	0.580	0.605	0.365	0.433
	GPT-4o-mini	0.700	0.621	0.446	0.503
	Few-Shot				
Barishal	Gemini 2.5 Flash	0.960	0.973	0.933	0.950
	Claude 3.7 Sonnet	0.920	0.921	0.886	0.901
	GPT-4o-mini	**0.98**	**0.98**	**0.98**	**0.98**
Chattogram	Gemini 2.5 Flash	0.940	0.939	0.941	0.940
	Claude 3.7 Sonnet	0.860	0.860	0.857	0.859
	GPT-4o-mini	**0.980**	**0.979**	**0.981**	**0.980**
Sylhet	Gemini 2.5 Flash	0.960	0.958	0.958	0.958
	Claude 3.7 Sonnet	0.940	0.635	0.634	0.633
	GPT-4o-mini	**0.98**	**0.98**	**0.98**	**0.98**
Noakhali	Gemini 2.5 Flash	0.940	0.942	0.937	0.939
	Claude 3.7 Sonnet	0.920	0.638	0.609	0.623
	GPT-4o-mini	**1.000**	**1.000**	**1.000**	**1.000**

Table 2: Sentiment analysis performance by dialect (native Bangla script).

Dialect	Model	Acc	Prec	Rec	F1
	Zero-Shot (Transliterated)				
Barishal	Gemini 2.5 Flash	0.840	0.667	0.527	0.585
	Claude 3.7 Sonnet	0.640	0.558	0.356	0.412
	GPT-4o-mini	0.900	0.640	0.556	0.586
Chattogram	Gemini 2.5 Flash	0.640	0.521	0.419	0.457
	Claude 3.7 Sonnet	0.580	0.455	0.375	0.390
	GPT-4o-mini	0.860	0.598	0.570	0.582
Sylhet	Gemini 2.5 Flash	0.560	0.587	0.403	0.463
	Claude 3.7 Sonnet	0.540	0.539	0.406	0.422
	GPT-4o-mini	0.860	0.609	0.591	0.592
Noakhali	Gemini 2.5 Flash	0.720	0.621	0.458	0.509
	Claude 3.7 Sonnet	0.640	0.589	0.404	0.453
	GPT-4o-mini	0.840	0.655	0.545	0.587
	Few-Shot (Transliterated)				
Barishal	Gemini 2.5 Flash	0.980	0.969	0.986	0.977
	Claude 3.7 Sonnet	0.920	0.624	0.603	0.613
	GPT-4o-mini	**0.98**	**0.98**	**0.98**	**0.98**
Chattogram	Gemini 2.5 Flash	0.660	0.630	0.442	0.518
	Claude 3.7 Sonnet	0.760	0.775	0.749	0.750
	GPT-4o-mini	**0.98**	**0.98**	**0.98**	**0.98**
Sylhet	Gemini 2.5 Flash	0.760	0.612	0.504	0.552
	Claude 3.7 Sonnet	0.860	0.593	0.585	0.585
	GPT-4o-mini	**0.98**	**0.98**	**0.98**	**0.98**
Noakhali	Gemini 2.5 Flash	0.960	0.959	0.959	0.959
	Claude 3.7 Sonnet	0.840	0.595	0.552	0.571
	GPT-4o-mini	**0.98**	**0.98**	**0.98**	**0.98**

Table 3: Sentiment analysis performance by dialect (transliterated text).

B Key Observations by Dialect

B.1 Barishal Dialect

GPT-4o-mini achieved near perfect sentiment classification scores (accuracy, precision, recall, and F1 all at 0.98) in few-shot conditions for both native and transliterated text. For translation tasks, Claude 3.7 Sonnet demonstrated the strongest zero-shot performance with transliteration, achieving a BLEU score of 0.424 and a METEOR score of 0.695. Among open-source models, Mistral-7B performed best with 0.865 accuracy in zero-shot sentiment analysis.

B.2 Chattogram Dialect

This dialect presented the most challenging results across all models. Closed-source models showed moderate performance in zero-shot settings, with GPT-4o-mini reaching 0.760 accuracy. Few-shot prompting improved results significantly, with GPT-4o-mini achieving 0.980 accuracy (near-perfect). Open-source models particularly struggled with Chattogram, with the best performance being 0.678 accuracy from Mistral-7B in few-shot transliteration mode. Translation scores remained consistently lower than other dialects across all conditions.

B.3 Sylhet Dialect

Claude 3.7 Sonnet achieved the highest zero-shot translation scores for this dialect with transliteration (BLEU: 0.431, METEOR: 0.692), demonstrating strong lexical and semantic understanding. GPT-4o-mini maintained perfect sentiment classification in few-shot settings (1.000 across all metrics). The dialect showed interesting patterns where zero-shot translation with transliteration substantially outperformed few-shot approaches, suggesting that contextual examples may introduce confusion for this particular dialectal variation.

B.4 Noakhali Dialect

Consistent with the other three dialects, GPT-4o-mini achieved perfect few-shot sentiment classification scores. Translation proved challenging across all models, with the highest BLEU score being 0.251 from GPT-4o-mini in zero-shot transliteration mode. Open-source models showed particularly poor performance, with Llama-3.1-8B achieving only 0.300 accuracy in zero-shot sentiment analysis. The dialect's linguistic distance from standard Bangla appears to pose significant challenges for all model architectures, particularly for open-source alternatives with limited Bengali representation in their training data.

Dialect	Model	Acc	Prec	Rec	F1
Barishal	Mistral-7B (Zero)	0.865	0.820	0.793	0.805
Barishal	Qwen-2.5-7B (Few+Trans)	0.851	0.799	0.774	0.786
Chattogram	Gemma-3-12B (Zero)	0.624	0.590	0.599	0.590
Chattogram	Mistral-7B (Few+Trans)	0.678	0.622	0.616	0.618
Sylhet	Qwen-2.5-7B (Zero)	0.647	0.610	0.662	0.596
Noakhali	Llama-3.1-8B (Zero)	0.300	0.304	0.314	0.196

Table 4: Open-Source Model Sentiment Analysis - Selected Best Performers

Dialect	Model	Setting	BLEU	METEOR	R1	R2	RL
		Zero-Shot					
Barishal	Gemini 2.5 Flash	Zero-Shot	0.013	0.136	0.231	0.179	0.211
	Claude 3.7 Sonnet	Zero-Shot	0.063	0.202	0.749	0.568	0.717
	GPT-4o-mini	Zero-Shot	0.022	0.165	0.431	0.311	0.400
Chattogram	Gemini 2.5 Flash	Zero-Shot	0.008	0.070	0.201	0.139	0.177
	Claude 3.7 Sonnet	Zero-Shot	0.057	0.153	0.724	0.502	0.668
	GPT-4o-mini	Zero-Shot	0.022	0.130	0.481	0.313	0.427
Sylhet	Gemini 2.5 Flash	Zero-Shot	0.008	0.073	0.092	0.075	0.089
	Claude 3.7 Sonnet	Zero-Shot	0.090	0.267	0.743	0.558	0.702
	GPT-4o-mini	Zero-Shot	0.028	0.177	0.449	0.321	0.417
Noakhali	Gemini 2.5 Flash	Zero-Shot	0.008	0.070	0.280	0.211	0.261
	Claude 3.7 Sonnet	Zero-Shot	0.047	0.131	0.691	0.496	0.655
	GPT-4o-mini	Zero-Shot	0.022	0.106	0.432	0.298	0.399
		Few-Shot					
Barishal	Gemini 2.5 Flash	Few-Shot	0.080	0.287	0.531	0.420	0.506
	Claude 3.7 Sonnet	Few-Shot	0.041	0.200	0.561	0.429	0.539
	GPT-4o-mini	Few-Shot	0.036	0.204	0.542	0.410	0.515
Chattogram	Gemini 2.5 Flash	Few-Shot	0.029	0.185	0.428	0.306	0.378
	Claude 3.7 Sonnet	Few-Shot	0.035	0.155	0.603	0.418	0.555
	GPT-4o-mini	Few-Shot	0.036	0.169	0.551	0.376	0.501
Sylhet	Gemini 2.5 Flash	Few-Shot	0.057	0.309	0.459	0.365	0.427
	Claude 3.7 Sonnet	Few-Shot	0.077	0.265	0.547	0.397	0.510
	GPT-4o-mini	Few-Shot	0.049	0.238	0.531	0.386	0.500
Noakhali	Gemini 2.5 Flash	Few-Shot	0.039	0.165	0.527	0.390	0.493
	Claude 3.7 Sonnet	Few-Shot	0.029	0.135	0.532	0.378	0.496
	GPT-4o-mini	Few-Shot	0.043	0.128	0.540	0.386	0.510

Table 5: Translation Performance on Native Bangla Script

Dialect	Model	Setting	BLEU	METEOR	R1	R2	RL
		Zero-Shot with Transliteration					
Barishal	Gemini 2.5 Flash	Zero-Shot	0.222	0.383	0.557	0.451	0.530
	Claude 3.7 Sonnet	Zero-Shot	0.424	0.695	0.904	0.846	0.894
	GPT-4o-mini	Zero-Shot	0.380	0.697	0.927	0.877	0.925
Chattogram	Gemini 2.5 Flash	Zero-Shot	0.202	0.400	0.505	0.462	0.502
	Claude 3.7 Sonnet	Zero-Shot	0.310	0.505	0.830	0.727	0.812
	GPT-4o-mini	Zero-Shot	0.252	0.479	0.854	0.758	0.841
Sylhet	Gemini 2.5 Flash	Zero-Shot	0.230	0.528	0.477	0.437	0.475
	Claude 3.7 Sonnet	Zero-Shot	0.431	0.692	0.880	0.816	0.868
	GPT-4o-mini	Zero-Shot	0.385	0.664	0.905	0.840	0.900
Noakhali	Gemini 2.5 Flash	Zero-Shot	0.143	0.345	0.581	0.491	0.561
	Claude 3.7 Sonnet	Zero-Shot	0.205	0.442	0.814	0.710	0.797
	GPT-4o-mini	Zero-Shot	0.251	0.488	0.857	0.767	0.847
		Few-Shot with Transliteration					
Barishal	Gemini 2.5 Flash	Few-Shot	0.130	0.343	0.663	0.530	0.630
	Claude 3.7 Sonnet	Few-Shot	0.109	0.264	0.653	0.515	0.630
	GPT-4o-mini	Few-Shot	0.082	0.276	0.634	0.482	0.604
Chattogram	Gemini 2.5 Flash	Few-Shot	0.034	0.178	0.506	0.362	0.456
	Claude 3.7 Sonnet	Few-Shot	0.048	0.154	0.670	0.465	0.615
	GPT-4o-mini	Few-Shot	0.054	0.178	0.603	0.410	0.548
Sylhet	Gemini 2.5 Flash	Few-Shot	0.086	0.331	0.591	0.486	0.558
	Claude 3.7 Sonnet	Few-Shot	0.073	0.260	0.606	0.453	0.568
	GPT-4o-mini	Few-Shot	0.094	0.254	0.632	0.476	0.601
Noakhali	Gemini 2.5 Flash	Few-Shot	0.039	0.173	0.562	0.426	0.532
	Claude 3.7 Sonnet	Few-Shot	0.035	0.115	0.628	0.452	0.591
	GPT-4o-mini	Few-Shot	0.043	0.121	0.629	0.456	0.590

Table 6: Translation Performance on Transliterated (Bangla) Script

Dialect	Model	BLEU	METEOR	R1	RL
Barishal	Mistral-7B (Zero)	0.067	0.274	0.596	0.566
Barishal	Gemma-3-12B (Zero)	0.048	0.136	0.712	0.653
Chattogram	Gemma-3-12B (Zero)	0.033	0.085	0.627	0.545
Chattogram	Mistral-7B (Few+Trans)	**0.123**	**0.334**	**0.739**	**0.705**
Sylhet	Qwen-2.5-7B (Zero)	0.037	0.075	0.616	0.547
Sylhet	Llama-3.1-8B (Few+Trans)	0.106	0.242	0.691	0.648
Noakhali	Llama-3.1-8B (Zero)	0.082	0.175	0.586	0.531
Noakhali	Gemma-3-12B (Few+Trans)	0.061	0.136	0.679	0.636

Table 7: Open-Source Model Translation Performance - Selected Best Performers

Prompt Used for Zero-Shot Translation

Prompt for Zero Shot Translation

You are an expert linguist specializing in Bengali dialects and language translation. You have extensive knowledge of various Bengali dialects including Barishal, Chattogram, Sylhet, and Noakhali dialects, and their relationship to standard Bengali.

Your task is to translate {dialect_name} dialect text to standard Bengali while preserving the original meaning, context, and emotional tone.

CRITICAL: You must respond with ONLY the translated text. Nothing else. No explanations. No additional words. No English text. Just the translation.

Now translate this {dialect_name} dialect text to standard Bengali:

{dialect_name} dialect text: {dialect_text}

Standard Bengali translation:

Prompt Used for Zero-Shot Translation (with Transliteration)

Prompt for Zero Shot Translation with Transliteration

You are an expert linguist specializing in Bengali dialects and language translation. You have extensive knowledge of various Bengali dialects including Barishal, Chattogram, Sylhet, and Noakhali dialects, and their relationship to standard Bengali.

Your task is to translate {dialect_name} dialect text to standard Bengali while preserving the original meaning, context, and emotional tone.

CRITICAL: You must respond with ONLY the translated text. Nothing else. No explanations. No additional words. No English text. Just the translation.

Now translate this {dialect_name} dialect text to standard Bengali:

{dialect_name} dialect text: {dialect_text}

Transliterated (Roman letters) {dialect_name} dialect text: {translit_dialect_text}

Standard Bengali translation:

Prompt Used for Zero-Shot Sentiment

Prompt for Zero Shot Sentiment

You are an expert in sentiment analysis for Bengali text, including both standard Bengali and various dialects. You have deep understanding of emotional nuances, cultural context, and linguistic patterns in Bengali language.

Your task is to classify the sentiment of the given {dialect_name} dialect text as either POSITIVE or NEGATIVE.

CRITICAL: You must respond with ONLY ”POSITIVE” or ”NEGATIVE”. Nothing else. No explanations. No additional words. Just the sentiment.

Now classify the sentiment of this {dialect_name} dialect text:

{dialect_name} dialect text: {dialect_text}

Sentiment (POSITIVE or NEGATIVE):

Prompt Used for Zero-Shot Sentiment (with Transliteration)

Prompt for Zero Shot Sentiment with Transliteration

You are an expert in sentiment analysis for Bengali text, including both standard Bengali and various dialects. You have deep understanding of emotional nuances, cultural context, and linguistic patterns in Bengali language.

Your task is to classify the sentiment of the given {dialect_name} dialect text as either POSITIVE or NEGATIVE.

CRITICAL: You must respond with ONLY ”POSITIVE” or ”NEGATIVE”. Nothing else. No explanations. No additional words. Just the sentiment.

Now classify the sentiment of this {dialect_name} dialect text:

{dialect_name} dialect text: {dialect_text}

Transliterated (Roman letters) {dialect_name} dialect text: {translit_dialect_text}

Sentiment (POSITIVE or NEGATIVE):

Prompt Used for Few-Shot Translation

Prompt for Few Shot Translation

You are an expert linguist specializing in Bengali dialects and language translation. You have extensive knowledge of various Bengali dialects, including Barishal, Chattogram, Sylhet, and Noakhali dialects, and their relationship to standard Bengali.

Your task is to translate {dialect_name} dialect text to standard Bengali while preserving the original meaning, context, and emotional tone.

CRITICAL: You must respond with ONLY the translated text. Nothing else. No explanations. No additional words. No English text. Just the translation.

EXAMPLES OF CORRECT TRANSLATIONS:

Dialect: "মোরা বরিশাইল্লা" -> Standard Bengali: "আমরা বরিশালের"
Dialect: "হুম মোগো বরিশাল" -> Standard Bengali: "হ্যাঁ, আমাদের বরিশাল"
Dialect: "মইনা বেশি কথা কইনা" -> Standard Bengali: "আমি বেশি কথা বলি না"
Dialect: "একদম পালতু বিরানি" -> Standard Bengali: "একদম ফালতু বিরিয়ানি"

Now translate this {dialect_name} dialect text to standard Bengali:

{dialect_name} dialect text: {dialect_text}

Standard Bengali translation:

Prompt Used for Few-Shot Translation (with Transliteration)

Prompt for Few Shot Translation with Transliteration

You are an expert linguist specializing in Bengali dialects and language translation. You have extensive knowledge of various Bengali dialects, including Barishal, Chattogram, Sylhet, and Noakhali dialects, and their relationship to standard Bengali.

Your task is to translate {dialect_name} dialect text to standard Bengali while preserving the original meaning, context, and emotional tone.

CRITICAL: You must respond with ONLY the translated text. Nothing else. No explanations. No additional words. No English text. Just the translation.

EXAMPLES OF CORRECT TRANSLATIONS:

Dialect: "মোরা বরিশাইল্লা" -> Standard Bengali: "আমরা বরিশালের"
Dialect: "হুম মোগো বরিশাল" -> Standard Bengali: "হ্যাঁ, আমাদের বরিশাল"
Dialect: "মইনা বেশি কথা কইনা" -> Standard Bengali: "আমি বেশি কথা বলি না"
Dialect: "একদম পালতু বিরানি" -> Standard Bengali: "একদম ফালতু বিরিয়ানি"

Now translate this {dialect_name} dialect text to standard Bengali:

{dialect_name} dialect text: {dialect_text}

Transliterated (Roman letters) {dialect_name} dialect text: {translit_dialect_text}

Standard Bengali translation:

Prompt Used for Few-Shot Sentiment

Prompt for Few Shot Sentiment

You are an expert in sentiment analysis for Bengali text, including both standard Bengali and various dialects. You have deep understanding of emotional nuances, cultural context, and linguistic patterns in Bengali language.

Your task is to classify the sentiment of the given {dialect_name} dialect text as either POSITIVE or NEGATIVE.

CRITICAL: You must respond with ONLY "POSITIVE" or "NEGATIVE". Nothing else. No explanations. No additional words. Just the sentiment.

EXAMPLES OF CORRECT CLASSIFICATIONS:

```
Text: "আমাদের বরিশাল নিয়ে আমাদের গর্ব" -> Sentiment: POSITIVE
Text: "একদম ফালতু বিরিয়ানি" -> Sentiment: NEGATIVE
Text: "সুন্দর হয়েছে" -> Sentiment: POSITIVE
Text: "ভাইয়ের বাড়ি এখন কোথায়" -> Sentiment: NEGATIVE
Text: "চিটাগং এর মেয়ে কেন চিটাগং এর হতে পারে না" -> Sentiment: POSITIVE
```

Now classify the sentiment of this {dialect_name} dialect text:

{dialect_name} dialect text: {dialect_text}

Sentiment (POSITIVE or NEGATIVE):

Prompt Used for Few-Shot Sentiment (with Transliteration)

Prompt for Few Shot Sentiment with Transliteration

You are an expert in sentiment analysis for Bengali text, including both standard Bengali and various dialects. You have deep understanding of emotional nuances, cultural context, and linguistic patterns in Bengali language.

Your task is to classify the sentiment of the given {dialect_name} dialect text as either POSITIVE or NEGATIVE.

CRITICAL: You must respond with ONLY "POSITIVE" or "NEGATIVE". Nothing else. No explanations. No additional words. Just the sentiment.

EXAMPLES OF CORRECT CLASSIFICATIONS:

```
Text: "আমাদের বরিশাল নিয়ে আমাদের গর্ব" -> Sentiment: POSITIVE
Text: "একদম ফালতু বিরিয়ানি" -> Sentiment: NEGATIVE
Text: "সুন্দর হয়েছে" -> Sentiment: POSITIVE
Text: "ভাইয়ের বাড়ি এখন কোথায়" -> Sentiment: NEGATIVE
Text: "চিটাগং এর মেয়ে কেন চিটাগং এর হতে পারে না" -> Sentiment: POSITIVE
```

Now classify the sentiment of this {dialect_name} dialect text:

{dialect_name} dialect text: {dialect_text}

Transliterated (Roman letters) {dialect_name} dialect text: {translit_dialect_text}

Sentiment (POSITIVE or NEGATIVE):

Robustness of LLMs to Transliteration Perturbations in Bangla

Fabiha Haider[1 *], **Md Farhan Ishmam**[2 *], **Fariha Tanjim Shifat**[1,3],
Md Tasmim Rahman[1], **Md Fahim**[1,4 †], **Md Farhad Alam Bhuiyan**[1]

[1]*Penta Global Limited* [2]*University of Utah* [3]*Missouri S&T*
[4]*CCDS, Independent University, Bangladesh*

[*]Equal Contribution [†]Project Lead

`{fabihahaider4,farhan.ishmam,fahimcse381}@gmail.com`

Abstract

Bangla text on the internet often appears in mixed scripts that combine native Bangla characters with their Romanized transliterations. To ensure practical usability, language models should be robust to naturally occurring script mixing. Our work investigates the robustness of current LLMs and Bangla language models under various transliteration-based textual perturbations, *i.e.*, we augment portions of existing Bangla datasets using transliteration. Specifically, we replace words and sentences with their transliterated text to emulate realistic script mixing, and similarly, replace the top k salient words to emulate adversarial script mixing. Our experiments reveal interesting behavioral insights and vulnerabilities to robustness in language models for Bangla, which can be crucial for deploying such models in real-world scenarios and enhancing their overall robustness. Our code is available at: `https://github.com/farhanishmam/BTL-Robustness`.

1 Introduction

In the digital era, Bangla is often written in its romanized form using English scripts due to the ubiquity of the QWERTY layout (Haider et al., 2024). With the growing popularity of Bangla keyboard layouts, particularly among mobile users, Bangla-English mixed script texts have become more common. This phenomenon is known as script-mixing, where multiple scripts are used in a single piece of text (Srivastava et al., 2020).

The current generation of Large Language Models (LLMs) has also excelled in tasks on transliterated or romanized Bangla (Fahim et al., 2024). However, their robustness to textual perturbations in Bangla has yet to be evaluated. Textual perturbation refers to any form of change or modification to the input text that can potentially impact the model's performance in a given task (Li et al., 2020a). Such perturbations can emulate realistic conditions (Moradi and Samwald, 2021) (e.g., removal or replacement of a word) or adversarial conditions (Li et al., 2018) (e.g., removal of most salient tokens (Raiyan et al., 2025)). Our work explores a form of replacement-based perturbation where words or sentences in the original Bangla scripts are replaced by their transliterations.

Current datasets in Bangla are limited to a single script, either in Bangla (Hasan et al., 2020; Islam et al., 2021) or English (Fahim et al., 2024). While code-mixed texts have been a topic of interest, where Bangla and English words are mixed, the datasets are usually limited to the English scripts (Alam et al., 2024). Evaluation of LLMs under script mixing can be crucial for deploying the model in realistic scenarios. We hence propose a scalable augmentation strategy to produce script-mixed text in Bangla and evaluate the robustness of models against such forms of perturbations. Our contributions can be summarized as:

- We present the first study to evaluate LLMs in three Bangla transliteration-based perturbations encompassing both realistic and adversarial settings.
- Our augmentation framework can be used to produce text that emulates script-mixing in Bangla at scale.
- Our experiments on a rich suite of closed-sourced and open-sourced LLMs, as well as Bangla language models, highlight the robustness vulnerability in Bangla.

2 Related Work

2.1 Textual Perturbation

Textual perturbations are either formulated as adversarial attacks that exploit the vulnerability of a system using an input, often tailored to that particular model (Li et al., 2020a), or common

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 338–346
December 23, 2025 ©2025 Association for Computational Linguistics

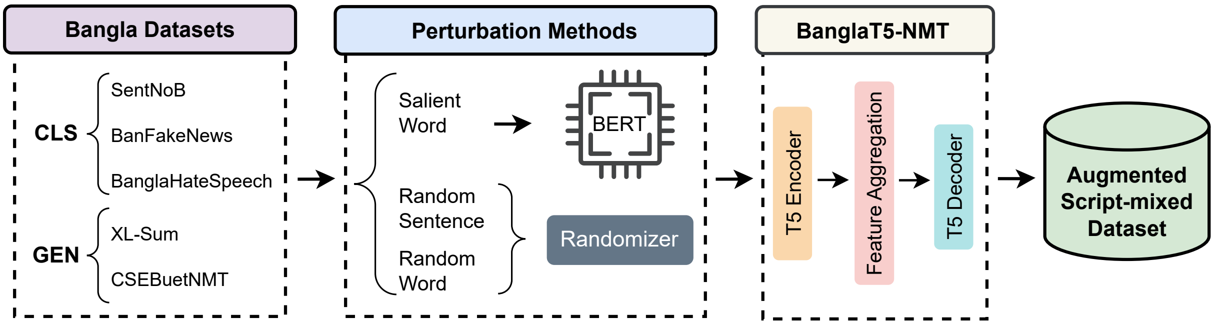


Figure 1: High level overview of our dataset perturbation pipeline.

perturbations that are typically encountered by texts in realistic scenarios (Moradi and Samwald, 2021). Adversarial perturbations saw some earlier success with rule-based methods, *e.g.*, synonym replacement, in both black-box and white-box settings (Jin et al., 2020; Alzantot et al., 2018). Few methods relied on using language models to generate adversarial examples (Li et al., 2020b; Garg and Ramakrishnan, 2020).

Realistic textual perturbations include character and word-level perturbations, *e.g.*, insertion, deletion, and replacement, which are used to simulate realistic errors in text (Moradi and Samwald, 2021; Le et al., 2022). Ours similarly uses word and sentence-level transliteration as realistic perturbations to simulate script mixing in text. We also experiment with the transliteration of the most salient word as a form of adversarial perturbation.

2.2 Robustness of Language Models

The robustness of language models refers to their inherent ability to sustain performance when exposed to input variations (Morris et al., 2020). While such studies on robustness are prevalent in English (Moradi and Samwald, 2021; Li et al., 2020a), the challenge is exacerbated in multilingual and low-resource contexts (Kaing et al., 2024). Robustness also refers to the language model's generalization capabilities under distribution shifts (Hendrycks et al., 2020). Our study focuses on evaluating this robustness in low-resource contexts, specifically examining the Bangla language under script distribution shifts.

2.3 Transliteration, Code-mixing, and Script-mixing

Transliterated texts, where native words are represented in foreign scripts, have been common in Indic languages through romanization (Madhani et al., 2023). This phenomenon is particularly prevalent in Bangla, where romanized scripts are used to write Bangla text. Current language models have shown strong performance on back-transliteration, *i.e.*, producing the original Bangla text from transliterated input (Fahim et al., 2024). Several downstream tasks have been explored on transliterated Bangla, including sentiment analysis (Hassan et al., 2016) and hate speech detection (Haider et al., 2024).

A closely related setting is code-switching or code-mixing, where words from multiple languages appear in the same text using different scripts (Sheth et al., 2025). Code-switching between Bangla and English is particularly common among Bangla speakers (Alam et al., 2024), though LLMs have shown degraded performance on such code-switched text (Mohamed et al., 2025). Our work differs in that we evaluate the robustness through Bangla dataset augmentations that mimic script-mixing (Srivastava et al., 2020), *i.e.*, multiple scripts coexist within the same text block.

3 Methodology

Our framework involves applying three types of perturbations to popular Bangla classification and generation datasets, as shown in Fig. 1.

3.1 Textual Perturbation

Each perturbation $p \in \mathcal{P}$ is defined as a function $p : \mathcal{T} \rightarrow \mathcal{T}^{\text{tr}}$, that takes an input text in native Bangla scripts B to produce text in transliterated scripts. For a model f, we quantify the average case performance as robustness over the test set distribution $\mathcal{D}$ (Hendrycks and Dietterich, 2019; Ishmam et al., 2025),

$$\mathbb{E}_{p\sim\mathcal{P}}[\mathbb{P}_{(B,y)\sim\mathcal{D}}((f(p(B)) = y)].$$

Dataset	SentNob	BanFakeNews	BanglaHateSpeech	XL-Sum	CSEBuetNMT
Total Samples	12k	49k	30K	8k	2.7M
Evaluated Samples	1568	2000	750	1012	1000
Vocab Size	24K	415K	64K	226K	1.3M
Min Word Length	3	1	1	7	1
Max Word Length	93	4650	537	3726	8353
Min Sentence Length	1	1	1	1	1
Max Sentence Length	20	679	78	370	262

Table 1: Statistics and number of samples taken for evaluation from the evaluation datasets.

The textual perturbation is implemented as a function that takes a slice of the input text and passes it to a transliteration model $f^{\text{tr}} : \mathcal{T} \rightarrow \mathcal{T}^{\text{tr}}$ to produce the transliterated text. The slicing of the text differs for each perturbation and has been defined in the latter sections.

3.1.1 Random Word Perturbation

For each word w_i in an input text $B = \{w_1, w_2, \dots w_n\} \in \mathcal{T}$ and a random word-level mask vector,

$$M = \{m_1, m_2, \dots, m_n\}, \quad m_i \sim \text{Bernoulli}(p),$$

where $p \in [0, 1]$ is the probability of perturbing a word, the random word perturbation can be defined:

$$p_{\text{rw}}(w_i) = \begin{cases} f^{\text{tr}}(w_i), & \text{if } m_i = 1, \\ w_i, & o/w. \end{cases} \quad (1)$$

3.1.2 Random Sentence Perturbation

Similar to §3.1.1, the sentence perturbation segments the input text B into sentences, $B = \{w_{1:i_1}, w_{i_1+1:i_2}, \dots w_{i_{n-1}+1:n}\} \in \mathcal{T}$, and uses sentence-level mask vectors. Following Eq.1, we define random sentence perturbation,

$$p_{\text{rs}}(w_{i:j}) = \begin{cases} f^{\text{tr}}(w_{i:j}), & \text{if } m_i = 1, \\ w_{i:j}, & o/w. \end{cases} \quad (2)$$

3.1.3 Salient Word Perturbation

Let $s_i = S(w_i, B)$ denote the saliency score assigned to word w_i, measuring its influence on the model's output. We calculate the saliency scores by averaging the attention scores of a BanglaBERT model (Bhattacharjee et al., 2022) across the sequence length, heads, and layers. We define a proportion p, and organize the words based on the descending order of saliency scores. We now define the set of top-p salient word indices,

$$\mathcal{I}_{sal} = \{i | s_i \text{ is among top } p \text{ scores}\}.$$

The salient word perturbation can be similarly defined as:

$$p_{\text{sal}}(w_i) = \begin{cases} f^{\text{tr}}(w_i), & \text{if } i \in \mathcal{I}_{sal}, \\ w_i, & o/w. \end{cases} \quad (3)$$

For each perturbation, the probability of perturbation p is taken as 20%. We use the BanglaT5_NMT model (Bhattacharjee et al., 2023) fine-tuned on the BanglaTLit dataset (Fahim et al., 2024) as our transliteration model f^{tr}.

3.2 Tasks & Datasets

We evaluate on five tasks: machine translation with CSEBuetNMT dataset (Hasan et al., 2020), hate speech detection with BanglaHateSpeech dataset (Romim et al., 2021), sentiment analysis with Sent-Nob dataset (Islam et al., 2021), fake news detection with BanFakeNews dataset (Hossain et al., 2020), and text summarization with XL-Sum dataset (Hasan et al., 2021). The number of samples taken from each dataset and their statistics are provided in Tab.1.

3.3 Baselines

We evaluate closed-source models: Claude-3.5 Sonnet, and GPT-4o (Hurst et al., 2024), open-source models: Qwen-2.5 32B (Qwen et al., 2025), Llama-3 70B (Grattafiori et al., 2024), and the Bangla language models: BanglaBERT (Bhattacharjee et al., 2022) and BanglaT5 (Bhattacharjee et al., 2023).

3.4 Evaluation Metrics

For classification, we use the standard metrics: Macro-F1 (M-F1), Weighted-F1 (W-F1), and Accuracy (Acc). Similarly, for generation tasks, we use BLEU score, Brevity Penalty, and ROUGE-2-F1.

4 Experimental Results

We evaluate model robustness across three perturbation strategies on classification and generation tasks (Tables 2 and 3). Most models achieve peak

Dataset	Model														
	Claude-3.5 Sonnet			GPT-4o			Qwen-2.5 32B			Llama-3 70B			BanglaBERT		
	M-F1	W-F1	Acc	M-F1	W-F1	Acc	M-F1	W-F1	Acc	M-F1	W-F1	Acc	M-F1	W-F1	Acc
SentNob															
Bangla Text (Base)	63.90	66.30	66.19	64.37	66.53	65.83	56.78	56.79	56.57	45.07	45.16	48.18	45.80	48.16	49.50
Random Words	63.73	66.01	65.96	63.48	65.46	64.88	52.50	52.54	52.00	45.77	45.81	48.69	45.13	47.19	48.05
Δ Base	-0.17	-0.29	-0.23	-0.89	-1.07	-0.95	-4.28	-4.25	-4.57	+0.70	+0.65	+0.51	-0.67	-0.97	-1.45
Random Sentences	60.86	63.04	62.92	58.93	60.85	59.90	48.69	48.71	48.00	46.48	46.67	49.90	35.88	37.48	41.30
Δ Base	**-3.04**	**-3.26**	**-3.27**	**-5.44**	**-5.68**	**-5.93**	**-8.09**	**-8.08**	**-8.57**	+1.41	+1.51	+1.72	**-9.92**	**-10.68**	**-8.20**
Salient Words	63.82	65.88	65.78	61.99	63.96	63.18	49.94	50.01	50.00	40.18	40.21	45.23	44.42	46.63	47.48
Δ Base	-0.08	-0.42	-0.41	-2.38	-2.57	-2.65	-6.84	-6.78	-6.57	**-4.89**	**-4.95**	**-2.95**	-1.38	-1.53	-2.02
BanFakeNews															
Bangla Text (Base)	66.80	66.88	68.42	85.93	85.93	85.93	52.07	78.11	78.00	50.58	75.88	75.36	92.98	92.99	93.00
Random Words	61.00	59.20	59.71	84.91	84.91	84.94	48.54	72.80	72.45	55.10	82.64	82.47	32.71	31.79	48.60
Δ Base	-5.80	-7.68	-8.71	-1.02	-1.02	-0.99	-3.53	-5.31	-5.55	+4.52	+6.76	+7.11	**-60.27**	**-61.20**	**-44.40**
Random Sentences	57.24	58.00	61.82	85.76	85.76	85.80	51.30	76.97	76.77	53.55	80.33	80.16	50.09	49.54	57.60
Δ Base	-9.56	-8.88	-6.60	-0.17	-0.17	-0.13	-0.77	-1.14	-1.23	+2.97	+4.45	+4.80	-42.89	-43.45	-35.40
Salient Words	47.86	49.02	55.00	84.89	84.89	84.90	47.55	71.33	71.00	52.55	78.82	78.74	33.14	32.23	48.80
Δ Base	**-18.94**	**-17.86**	**-13.42**	**-1.04**	**-1.04**	**-1.03**	**-4.52**	**-6.78**	**-7.00**	**+1.97**	**+2.94**	**+3.38**	-59.84	-60.76	-44.20
BanglaHateSpeech															
Bangla Text (Base)	85.54	87.29	87.56	79.90	82.13	82.13	83.77	83.77	84.00	53.21	53.17	59.39	91.45	92.33	92.27
Random Words	85.94	87.48	87.67	78.04	80.23	80.00	80.91	80.91	81.00	51.96	51.93	58.38	87.16	88.25	88.00
Δ Base	+0.40	+0.19	+0.11	-1.86	-1.90	-2.13	-2.86	-2.86	-3.00	-1.25	-1.24	**-1.01**	-4.29	-4.08	-4.27
Random Sentences	81.95	83.81	83.78	75.52	77.74	77.33	71.44	71.44	72.00	62.58	62.62	65.66	59.97	59.11	59.87
Δ Base	**-3.59**	**-3.48**	**-3.78**	**-4.38**	**-4.39**	**-4.80**	**-12.33**	**-12.33**	**-12.00**	+9.37	+9.45	+6.27	**-31.48**	**-33.22**	**-32.40**
Salient Words	84.52	86.04	86.10	77.59	79.90	79.73	76.89	76.89	77.00	51.70	51.70	58.60	76.89	77.96	77.33
Δ Base	-1.02	-1.25	-1.46	-2.31	-2.23	-2.40	-6.88	-6.88	-7.00	**-1.51**	**-1.47**	-0.79	-14.56	-14.37	-14.94

Table 2: Macro-F1(M-F1), Weighted-F1(W-F1), Accuracy(Acc) score for the classification tasks: sentiment analysis, fake news detection, and hate speech classification on SentNob, BanFakeNews, and BanglaHateSpeech, respectively. Gray indicates base/clean text performance, and **cyan** indicates worst performance degradation.

Dataset	Model														
	Claude-3.5 Sonnet			GPT-4o			Qwen-2.5 32B			Llama-3 70B			BanglaT5		
	BLEU	BP	R2-F1	BLEU	BP	R2-F1	BLEU	BP	R2-F1	BLEU	BP	R2-F1	BLEU	BP	R2-F1
XL-Sum															
Bangla Text (Base)	0.000	1.00	0.00	0.002	0.99	0.01	0.001	0.99	0.01	0.004	0.97	0.01	0.025	0.62	0.03
Random Words	0.000	1.00	0.00	0.002	0.98	0.00	0.002	0.99	0.01	0.003	0.98	0.01	0.016	0.59	0.02
Δ Base	**-0.00**	0.00	-	-0.00	-0.01	-	+0.00	0.00	-	-0.00	+0.01	-	-0.01	-0.03	-
Random Sentences	0.000	1.00	0.00	0.002	0.98	0.01	0.001	0.99	0.01	0.003	0.98	0.01	0.017	0.59	0.03
Δ Base	+0.00	0.00	-	-0.00	-0.01	-	+0.00	0.00	-	-0.00	+0.01	-	-0.01	-0.03	-
Salient Words	0.000	1.00	0.00	0.001	0.99	0.00	0.001	0.99	0.00	0.003	0.98	0.01	0.006	0.60	0.01
Δ Base	+0.00	0.00	-	**-0.00**	0.00	-	**-0.00**	0.00	-	**-0.00**	+0.01	-	**-0.02**	-0.02	-
CSEBuetNMT															
Bangla Text (Base)	0.215	0.94	0.223	0.215	0.96	0.220	0.171	0.99	0.191	0.059	0.95	0.073	0.241	0.93	0.233
Random Words	0.191	0.94	0.201	0.184	0.97	**0.195**	0.134	0.97	**0.150**	0.051	0.92	0.065	0.180	0.91	**0.185**
Δ Base	**-0.02**	0.00	-	-0.03	+0.01	-	-0.04	-0.02	-	**-0.01**	**-0.03**	-	-0.06	-0.02	-
Random Sentences	0.199	0.89	0.212	0.109	0.95	0.125	0.052	0.96	0.066	0.065	0.94	**0.079**	0.027	0.79	0.037
Δ Base	-0.02	**-0.05**	-	**-0.11**	**-0.01**	-	**-0.12**	**-0.03**	-	+0.01	-0.01	-	**-0.21**	**-0.14**	-
Salient Words	0.262	0.96	**0.251**	0.180	0.97	0.192	0.113	0.98	0.140	0.052	0.92	0.068	0.180	0.90	0.184
Δ Base	+0.05	+0.02	-	-0.03	+0.01	-	-0.06	-0.01	-	-0.01	-0.03	-	-0.06	-0.03	-

Table 3: BLEU score, BP: Brevity Penalty, R2-F1 for the translation and summarisation tasks on the CSEBuetNMT and XL-Sum datasets, respectively. Gray indicates base/clean text performance, and **cyan** indicates worst performance degradation. The difference between the R2-F1 scores is not calculated as it doesn't hold any meaningful value.

performance on clean text and show degradation under the perturbations.

4.1 Effect of Perturbation Techniques

Random sentence and salient word perturbations induce higher performance drops than random word perturbations. For instance, Claude-3.5 Sonnet shows a 3.27% accuracy drop on SentNob under random sentence perturbation versus only 0.23% for random word perturbation. The vulnerability also varies across tasks, *e.g.* random sentence perturbation is more challenging on

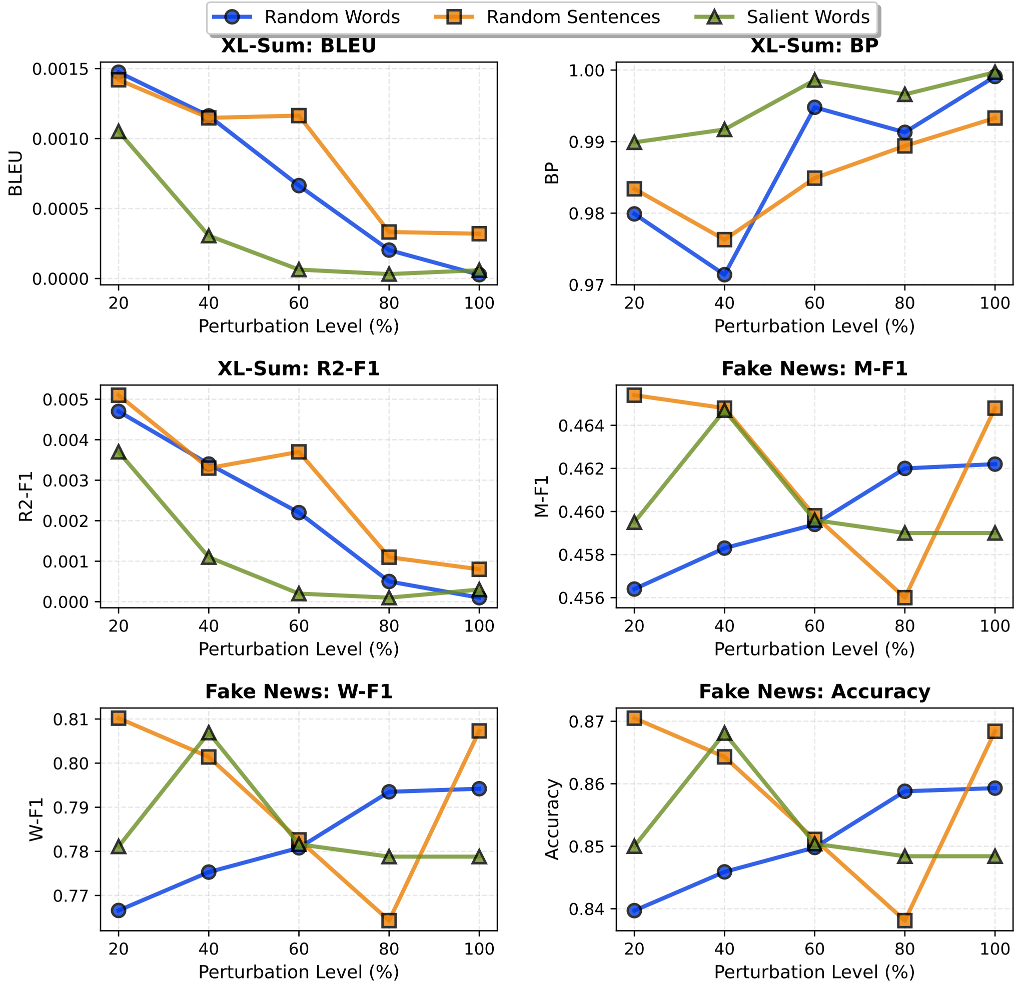


Figure 2: Impact of varying perturbation levels on the performance of the GPT-4o model in classification (Fake News) and generation (XL-Sum) tasks.

SentNob, BanglaHateSpeech, and CSEBuetNMT, while salient word perturbation is more severe on BanFakeNews and XL-Sum.

Smaller language models, like BanglaBERT and BanglaT5, show higher vulnerability, confirming their strong reliance on key lexical and semantic cues. Among LLMs, GPT-4o and Llama-3 exhibit relatively better robustness, maintaining smaller performance drops across all metrics, compared to Claude-3.5 Sonnet and Qwen-2.5. However, GPT-4o was less robust on generative tasks, *e.g.*, on the CSEBuetNMT dataset. We attribute the model-wise performance variance to the pretraining data distribution and exposure to code-mixed and script-mixed data during training.

4.2 Performance Degradation across Tasks

For classification tasks, LLMs showed relatively consistent degradation patterns: 3-8.5% on SentNob, 1-19% on BanFakeNews, and 1.5-12.5% on BanglaHateSpeech across all metrics. By contrast, BanglaBERT suffers dramatically larger drops, with F1-score degradation reaching 60% and accuracy declining by 44.5%. For generative tasks, the degradation was relatively higher for CSEBuetNMT than XL-sum, with BanglaT5 being more vulnerable than the LLMs.

4.3 Performance across Perturbation Levels

In Fig. 2, we observe a substantial decline in the summarization metrics BLEU and R2-F1, showing GPT-4o's vulnerability to increasing perturbation levels across all perturbation types. Random-word and salient-word perturbations show a consistent downward trend for the classification task. In contrast, random-sentence perturbation dips sharply at the 80% level, followed by an unexpected rebound at 100%. This suggests that the model becomes confused when only a small number of sentences are transliterated, whereas fully perturbed input allows it to settle into a more stable interpretation.

5 Discussion

We discuss the underlying causes of performance degradation in script-mixed scenarios, promising steps for mitigation, and other future directions.

5.1 The Tokenization Bottleneck

The substantial performance degradation observed in script-mixed texts can be largely attributed to fundamental limitations in tokenization. Firstly, the choice of tokenization method varies across models and can be an inherent limitation in script-mixing. For instance, models such as BERT and T5 employ WordPiece (Schuster and Nakajima, 2012) and SentencePiece (Kudo and Richardson, 2018) tokenization, respectively, which exhibit reduced robustness compared to the Byte Pair Encoding (BPE) (Gage, 1994) used in modern LLMs. The older tokenization methods struggle to maintain consistent granularity of mixed tokens, leading to suboptimal encoding.

Secondly, the process of tokenization itself constitutes an inherent architectural bottleneck, especially for cross-script processing. In script-mixed texts, using tokenizers trained predominantly on one script, typically Latin, penalizes foreign or untrained scripts (Land and Arnett, 2025). These tokenizers frequently fragment non-English tokens into excessive subword units or map them to rare and underrepresented vocabulary entries, occasionally resorting to unknown token markers. This phenomenon reflects a deeper issue of *vocabulary bias*, where tokenizers optimized on monolingual or Latin-script-dominant corpora show systematic disadvantages when processing alternative scripts, resulting in unnecessarily long token sequences and potential information loss at the encoding stage.

5.2 BLT and Multi-script Tokenizers

Byte Latent Transformers (BLT) (Pagnoni et al., 2025) have shown great empirical robustness to input perturbations and warrant investigation in script-mixing scenarios, as their byte-level processing naturally sidesteps script tokenizing limitations. Multilingual or transliteration-aware tokenizers with joint-script vocabularies offer a potential direct solution. Such tokenizers would require balancing the training data to ensure equitable representation across scripts and prevent the replication of existing script biases.

5.3 Script Normalization

A practical and easier approach to improve script-mixing robustness can be achieved through script normalization, *i.e.*, conversion of mixed scripts to a single script that is the most dominant throughout the input text. One option is to train a dedicated normalizer model, *e.g.*, a sequence-to-sequence model similar to BanglaT5-NMT (Fahim et al., 2024), but for script conversion. Alternatively, LLMs with reasoning capabilities could be prompted to normalize scripts in the thinking process first before proceeding with the task.

5.4 Can training improve robustness?

The language models can be either continually pre-trained or fine-tuned on the script-mixed dataset. Continual pre-training on multilingual or multi-script corpora should mitigate monoscript bias and enable models to learn robust cross-script correspondences. By exposing models to diverse script combinations during pre-training, we can potentially encode invariance to script perturbations directly into the model's representations. Task-specific fine-tuning on script-mixed text could also be a viable approach, but raises difficulty in estimating the distribution of scripts, leading to plausibly higher degradation due to overfitting.

5.5 Extension to Multimodal Settings

Our perturbation pipeline can be extended to multimodal scenarios, *e.g.*, visual question answering (Antol et al., 2015; Ishmam et al., 2025) on Bangla-regional images (Barua et al., 2025), which can investigate cross-visual perturbations, such as swapping cultural elements between images, or evaluating on script-mixed questions.

6 Conclusion

Our work evaluates LLMs and Bangla LMs under transliteration-based perturbations on random words, random sentences, and salient words. Our framework provides a scalable method for augmenting existing Bangla datasets to produce their script-mixed counterparts, thereby assessing the robustness of language models. Our findings reveal that discriminative models are vulnerable to script-mixing, whereas generative models are relatively more robust. We envision that our work will open doors for future research in Bangla script-mixing.

Limitations

Our study uses only transliteration-based perturbations, which are a subset of replacement-based perturbations. Other categories of perturbations, *e.g.,* insertion, deletion, and paraphrasing, haven't been explored and could provide a holistic view of the model's robustness. Our proposed robustness enhancement strategies have not been empirically verified and could be a potential future direction.

References

Sadia Alam, Md Farhan Ishmam, Navid Hasin Alvee, Md Shahnewaz Siddique, Md Azam Hossain, and Abu Raihan Mostofa Kamal. 2024. Bnsentmix: A diverse bengali-english code-mixed dataset for sentiment analysis. *arXiv preprint arXiv:2408.08964*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, Fabiha Haider, Fariha Tanjim Shifat, Md Tasmim Rahman Adib, Anam Borhan Uddin, Md Farhan Ishmam, and Md Farhad Alam. 2025. Chitrojera: A regionally relevant visual question answering dataset for bangla. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 473–491. Springer.

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.

Md Fahim, Fariha Tanjim Shifat, Fabiha Haider, Deeparghya Dutta Barua, MD Sakib Ul Rahman Sourove, Md Farhan Ishmam, and Md Farhad Alam Bhuiyan. 2024. BanglaTLit: A benchmark dataset for back-transliteration of Romanized Bangla. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14656–14672, Miami, Florida, USA. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, ... Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, and Md Farhad Alam. 2024. Banth: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M. Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2612–2623, Online. Association for Computational Linguistics.

Asif Hassan, Mohammad Rashedul Amin, Abul Kalam Al Azad, and Nabeel Mohammed. 2016. Sentiment analysis on bangla and romanized bangla text using deep recurrent models. In *2016 International Workshop on Computational Intelligence (IWCI)*, pages 51–56. IEEE.

Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online. Association for Computational Linguistics.

Md Zobaer Hossain, Md Ashraful Rahman, Md Saiful Islam, and Sudipta Kar. 2020. BanFakeNews: A dataset for detecting fake news in Bangla. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2862–2871, Marseille, France. European Language Resources Association.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Md Farhan Ishmam, Ishmam Tashdeed, Talukder Asir Saadat, Md Hamjajul Ashmafee, Abu Raihan Mostofa Kamal, and Md Azam Hossain. 2025. Visual robustness benchmark for visual question answering (vqa). In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6623–6633. IEEE.

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.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Hour Kaing, Chenchen Ding, Hideki Tanaka, and Masao Utiyama. 2024. Robust neural machine translation for abugidas by glyph perturbation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 311–318, St. Julian's, Malta. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Sander Land and Catherine Arnett. 2025. Bpe stays on script: Structured encoding for robust multilingual pretokenization. *arXiv preprint arXiv:2505.24689*.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. *arXiv preprint arXiv:2203.10346*.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Yash Madhani, Sushane Parthan, Priyanka Bedekar, Gokul Nc, Ruchi Khapra, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2023. Aksharantar: Open indic-language transliteration datasets and models for the next billion users. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 40–57.

Amr Mohamed, Yang Zhang, Michalis Vazirgiannis, and Guokan Shang. 2025. Lost in the mix: Evaluating llm understanding of code-switched text. *Preprint*, arXiv:2506.14012.

Milad Moradi and Matthias Samwald. 2021. Evaluating the robustness of neural language models to input perturbations. *arXiv preprint arXiv:2108.12237*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, et al. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Syed Rifat Raiyan, Md Farhan Ishmam, Abdullah Al Imran, and Mohammad Ali Moni. 2025. Frugalprompt: Reducing contextual overhead in large language models via token attribution. *arXiv preprint arXiv:2510.16439*.

Nauros Romim, Mosahed Ahmed, Hriteshwar Talukder, and Md. Saiful Islam. 2021. Hate speech detection in the bengali language: A dataset and its baseline evaluation. In *Proceedings of International Joint Conference on Advances in Computational Intelligence*, pages 457–468, Singapore. Springer Singapore.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Rajvee Sheth, Samridhi Raj Sinha, Mahavir Patil, Himanshu Beniwal, and Mayank Singh. 2025. Beyond monolingual assumptions: A survey of code-switched nlp in the era of large language models. *Preprint*, arXiv:2510.07037.

Abhishek Srivastava, Kalika Bali, and Monojit Choudhury. 2020. Understanding script-mixing: A case study of hindi-english bilingual twitter users. In *Proceedings of the 4th Workshop on Computational Approaches to Code Switching*, pages 36–44.

Generative Data Augmentation for Improving Semantic Classification

Shadman Rohan[1*] **Mahmud Elahi Akhter**[4*] **Ibraheem Muhammad Moosa**[3]
Nabeel Mohammed[2] **Amin Ahsan Ali**[1] **AKM Mahbubur Rahman**[1†]
[1]Center for Computational & Data Sciences [2]North South University, Bangladesh
[3]Pennsylvania State University, USA [4]Queen Mary University of London, UK
shadmanrohan@gmail.com m.akhter@qmul.ac.uk ibraheem.moosa@psu.edu
nabeel.mohammed@northsouth.edu {aminali, akmmrahman}@iub.edu.bd

Abstract

We study sentence-level generative data augmentation for Bangla semantic classification across four public datasets and three pretrained model families (BanglaBERT, XLM-Indic, mBERT). We evaluate two widely used, reproducible techniquesparaphrasing (mT5-based) and round-trip backtranslation (BnEnBn)and analyze their impact under realistic class imbalance. Overall, augmentation often helps, but gains are tightly coupled to label quality: paraphrasing typically outperforms backtranslation and yields the most consistent improvements for the monolingual model, whereas multilingual encoders benefit less and can be more sensitive to noisy minority-class expansions. A key empirical observation is that the neutral class appears to be a major source of annotation noise, which degrades decision boundaries and can cap the benefits of augmentation even when positive/negative classes are clean and polarized. We provide practical guidance for Bangla sentiment pipelines: (i) use simple sentence-level augmentation to rebalance classes when labels are reliable; (ii) allocate additional curation and higher inter-annotator agreement targets to the neutral class. Our results indicate when augmentation helps and suggest that data qualitynot model choice alonecan become the limiting factor.

1 Introduction

Semantic classification is an important task in natural language processing, facilitating the understanding of textual content by machines. Among the myriad languages worldwide, Bangla stands out as one of the most spoken, yet it encounters challenges related to data scarcity and resource limitation. It is important to note that, many efforts were undertaken to create datasets for sentiment analysis in Bangla but very few meet the rigorous quality benchmarks such as measuring Inter-Annotator Agreement (IAA). In the literature of data augmentation, word replacement with synonyms or nearest embedding has been a traditional strategy. However, its efficacy is limited, especially in the context of complex languages and nuanced semantics. On the other hand, generative data augmentation provides an avenue to produce more diverse and contextually relevant data samples. Paraphrasing and backtranslation emerge as notable generative strategies, with the latter being a staple in the machine translation domain.

In this study, we show how generative data augmentation can be used to improve the performance of sentiment classification. For this purpose we use four different datasets and three different models. We show our results for both multilingual and monolingual models. We found that monolingual models along with paraphrased data augmentation performed best, followed by backtranslation. Furthermore, the multilingual language models performance did not improve with data augmentation. While improving variation of good datapoints, we found that these techniques can also magnify the noise in the datasets. Additionally, we also found considerable label noise in the datasets and we empirically showed how this noise impeded the performance of sentiment classifiers. In this paper we study the potential of generative data augmentation, with a specific focus on its application in Bangla semantic classification.

Our goal is not to propose a new augmenta-

[*] Equal contribution.

[†] Corresponding author.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 347–356
December 23, 2025 ©2025 Association for Computational Linguistics

tion algorithm. Rather, we provide a controlled, Bangla-centric study across four public datasets and three pretrained model families (monolingual, Indic-multilingual, massively multilingual). We quantify when simple sentence-level augmentations help and, crucially, show that benefits are bounded by *label quality*, with the neutral class emerging as the primary source of noise. In summary, our contributions are primarily three-fold:

1. We investigate the viability of generative data augmentation for Bangla sentiment classification across four public Bangla social media datasets.

2. We find that paraphrasing works better as a generative data augmentation method compared to backtranslation and this effect is more prominent in monolingual language models.

3. We further identify that the neutral class is the main source of label noise in the chosen datasets.

2 Related Work

There have been numerous foundational works that have shaped the field of Sentiment Analysis (SA). One of the earliest notable researches was conducted by Hu and Liu (2004), where they presented the Aspect-Based Opinion Mining model, a predecessor of the Feature-Based Opinion Mining Model. Their work emphasized the nuances of customer reviews, providing an analytical framework for subsequent studies.

Benchmark datasets play a significant role in the evolution and validation of SA techniques. Prominent among them are the Stanford Sentiment Treebank (Socher et al., 2013), IMDB Movie Reviews Dataset (Maas et al., 2011), Amazon Product Data, and Sentiment140 (Go et al., 2009). Over the years, these datasets have served as standard benchmarking resources for sentiment analysis.

2.1 Sentiment Analysis in Bangla

When it comes to studies specific to Bangla, many prioritized the creation of clean datasets with extensive pre-processing, as seen in the works of Khatun and Rabeya (2022) and Islam et al. (2020). Furthermore, some studies, like Hossain et al. (2020), have even incorporated code-mixed samples integrating both Bangla and English. In contrast, others such as Islam et al. (2021) have made efforts to represent real-world data by incorporating noisy samples from social media platforms. In terms of classification, most researches have leaned towards a tri-class labeling system - positive, negative, and neutral. However, only a selected few, such as Islam et al. (2021), delved deeper by further dividing the positive and negative sentiments based on their intensity - weak or strong. Other techniques, such as the one presented by Abu Taher et al. (2018), involved N-Gram Based Sentiment Mining using Support Vector Machine. Chakraborty et al. (2022) explored a ternary sentiment classification for Bangla text using both Support Vector Machine and Random Forest Classifier.

Expanding the lexicon for sentiment analysis in Bangla was the primary goal for studies like (Naim, 2021) and (Bhowmik et al., 2022). An enriched vocabulary set has been observed to enhance the efficacy of sentiment models. Additionally, these studies employed aspect-based sentiment analysis techniques, a testament to the influence of Hu and Liu's foundational work. Some researches ventured into cross-lingual approaches, such as the work of Sazzed (2020), which involved translating Bangla sentences into English for sentiment analysis. However, the community did not wholly embrace this due to the dependencies it introduced. Furthermore, certain datasets were inhibited by class imbalances, as noted with Wahid et al. (2019), which had a disproportionate number of positive or negative samples compared to neutral ones. Recent works, including Islam et al. (2021) and Hossain et al. (2020), have not only emphasized the importance of data quality but also focused on the meticulousness of data annotation. Related Bangla resource creation includes BenCoref for coreference, highlighting annotation design and cross-domain coverage (Rohan et al., 2023).

Progressing further, Bhowmick and Jana (2021) applied transformer-based models like BERT and XLM-ROBERTA, signifying the continuous evolution and adaptability of sentiment analysis techniques in Bangla. Taking a more complex approach, Rafi-Ur-Rashid et al. (2022) employed an ensemble of deep learning models. This research also tackled class-imbalanced data using

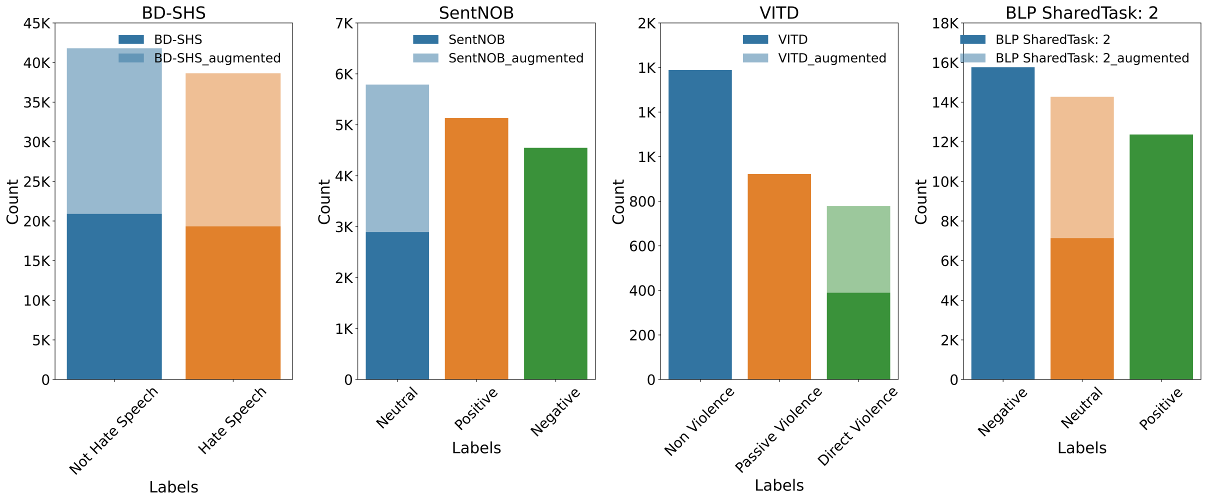


Figure 1: Comparison of dataset sizes before and after augmentation. The augmentation can be either backtranslation or paraphrasing.

the focal loss function. Despite the vast array of datasets available, many suffer from a domain bias. The SentNoB dataset (Islam et al., 2021) is currently considered the most robust, offering diversity across 13 domains and presenting a dataset with 15,000 Bangla samples.

3 Methodology & Experiment Details

We study two sentence-level augmentation methods, paraphrasing and back-translation, to increase minority-class coverage while preserving labels in Bangla sentiment analysis.

Traditional augmentation in NLP often relied on *synonym substitution* and *embedding-based replacement*. For temporal expressions, Kolomiyets et al. (2011) substituted tokens with WordNet (Miller et al., 1990) or model-predicted synonyms, assuming local replacements largely preserve meaning. Later, Kobayashi (2018) and Wei and Zou (2019) replaced words with nearest-neighbor or LM-suggested alternatives. However, in sentiment analysis even minimal lexical edits can flip polarity, and single-word substitutions frequently act like adversarial artifacts that distort semantics and inject label noise (Alzantot et al., 2018; Kaushik et al., 2020). Surveys further note that word-level noising yields limited *semantic* diversity and can harm label fidelity, recommending sentence-level methods when preservation matters (Feng et al., 2021). Accordingly, we exclude synonym/embedding replacement in our study.

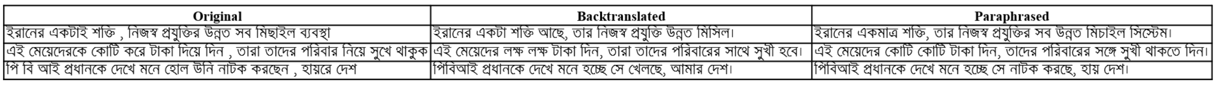
3.1 Paraphrasing

Paraphrase-based augmentation aims to increase lexical and syntactic diversity while preserving task-relevant semantics, mitigating overfitting to surface forms and reducing minority-class sparsity (Fadaee et al., 2017; Wei and Zou, 2019). We generate one paraphrase per source sentence using the mT5-based Bangla paraphrasing model from Akil et al. (2022), built on Xue et al. (2021). We do not apply post-generation filtering; given the reported semantic fidelity of this resource, capping at a single paraphrase limits distributional shift (Akil et al., 2022).

3.2 Back-translation

Back-translation rewrites a sentence by translating it to a pivot language and back, typically preserving meaning while introducing natural syntactic variation (Sennrich et al., 2016; Edunov et al., 2018; Xie et al., 2020). We employ Bangla→English→Bangla using the Bangla–English MT system of Hasan et al. (2020), choosing English as the pivot due to model maturity and availability. Although round-trip translation can introduce minor noise, the resulting variation is generally label-consistent and improves robustness to social-media artifacts (informality, misspellings, code-mixing) in our datasets (Edunov et al., 2018; Xie et al., 2020).

To keep the pipeline simple and reproducible-and to isolate the effect of class rebalancingwe intentionally avoid perplexity- or classifier-based filtering. Filtering/selection strategies (e.g.,



Original	Backtranslated	Paraphrased
ইরানের একটাই শক্তি , নিজস্ব প্রযুক্তির উন্নত সব মিছাইল ব্যবস্থা	ইরানের একটা শক্তি আছে, তার নিজস্ব প্রযুক্তি উন্নত মিসিল।	ইরানের একমাত্র শক্তি, তার নিজস্ব প্রযুক্তির সব উন্নত মিচাইল সিস্টেম।
এই মেয়েদেরকে কোটি করে টাকা দিয়ে দিন , তারা তাদের পরিবার নিয়ে সুখে থাকুক	এই মেয়েদের লক্ষ লক্ষ টাকা দিন, তারা তাদের পরিবারের সাথে সুখী হবে।	এই মেয়েদের কোটি কোটি টাকা দিন, তাদের পরিবারের সঙ্গে সুখী থাকতে দিন।
পি বি আই প্রধানকে দেখে মনে হোল উনি নাটক করছেন , হায়রে দেশ	পিবিআই প্রধানকে দেখে মনে হচ্ছে সে খেলছে, আমার দেশ।	পিবিআই প্রধানকে দেখে মনে হচ্ছে সে নাটক করছে, হায় দেশ।

Figure 2: Comparison of semantic preservation between paraphrased and backtranslated sentences from SentNoB dataset

Dataset	Text	Assigned	Correct
VITD	আপু আপনি একটু হিজাপ পরেন। ধন্যবাদ	Direct Violence	Non Violence
	সঠিক কাজ হয়েছে। শিক্ষা প্রতিষ্ঠান ধর্ম পালনের জায়গা না। এখানে শিক্ষা ও ভবিষ্যৎ গড়ার লক্ষ্যে আসে। বাংলাদেশের কোন হাইস্কুলে বোরকা পড়ার বিধান আছে?	Direct Violence	Passive Violence
	অভিযোগ আবার কেমন শব্দ নষ্টা? হুতাশে এই পেশায় চলে এসেছিস নাকি? গুলি করেছে বলবি!!	Non Violence	Direct Violence
	নাটক টা অনেক ভাল ছিল 🙄	Passive Violence	Non Violence
	মুসলিমদের জন্য আলাদা শিক্ষা প্রতিষ্ঠান করা হোক।।	Direct Violence	Non Violence
BLP Task 2	খুব ভালো সিদ্ধান্ত	Neutral	Positive
	পরিবেশবান্ধব পদ্ধতিতে জুমচাষের পরামর্শ বিশেষজ্ঞদের	Negative	Neutral
	আইপিএলের ইতিহাসে নতুন মাইলফলক স্থাপন করলেন কোহলি	Negative	Neutral
	দুঃখ জনক ঘটনা	Positive	Neutral
	করোনায় আরও ৩০ জনের মৃত্যু , মোট প্রাণহানি ৩১৮৪	Positive	Neutral
SentNoB	তুমি রেপারই হও , ডাক্তার হওয়ার দরকার নাই তোমার	Neutral	Negative
	এবার হিজরাদের নিয়ে রিপোর্ট করেন ওদের যন্ত্রণায় মানুষ অতিষ্ঠ হয়ে আছে	Positive	Negative

Figure 3: Examples of few incorrect labels we identified in the datasets. 'Assigned' shows the original label and 'Correct' shows our assessment of the appropriate label, illustrating label noise across all classes.

semantic-consistency thresholds, uncertainty-aware sampling) are orthogonal enhancements and left to future work.

3.3 Datasets

For our tasks, we utilized four datasets: BLP Shared Task 1: Violence Inciting Text Detection (VITD)(Saha et al., 2023), BLP Workshop Shared Task 2(Saha et al., 2023), SentNoB (Islam et al., 2021), and BD-SHS(Romim et al., 2022).

These datasets were selected to provide diverse evaluation scenarios for augmentation techniques. BD-SHS contains over 50,200 binary-labeled comments (hate/non-hate) from social media, serving as a control for binary classification against multi-class tasks. SentNoB comprises 15,000 manually annotated samples from social media across 13 domains with three-way sentiment labels (positive, negative, neutral), representing noisy informal Bangla text. VITD focuses on violence detection with three classes (direct violence, passive violence, non-violence) from YouTube comments about violent incidents in Bengal. BLP Task 2 involves three-class sentiment analysis of social media posts. This diversity in task types (hate speech detection, violence detection, sentiment analysis), class configurations (binary vs. ternary), and data characteristics (formal vs. noisy text) enables comprehensive assessment of how dataset properties affect augmentation efficacy.

These datasets serve as reference points for evaluating the effectiveness of the augmentation techniques. Here, BD-SHS serves more as a control for binary classification against the more complex multi-class classification. Apart from that, as stated earlier, we only augmented the minority class samples. For SentNoB and BLP Task 2, the minority class was the neutral class and for VITD it was the direct violence class (15%). Our augmentation results can be seen in Figure 1. Through our augmentation we mostly doubled the samples of minority class in order to improve the class imbalance of each of the datasets. We show the quality of backtranslation and paraphrased sentences in Figure 2. These sentences were taken from SentNoB dataset.

Additionally, as mentioned earlier, we found considerable label noise in the datasets from our qualitative analysis and we showcase some of these in Figure 3. We can see that samples across all classes are mislabeled and this is likely to impact the performance of models as this increases both false positive and false negative rates. We touch upon the impact of label noise on performance in Section 4.3.

Code and recipes. All augmentation scripts, preprocessing, and training configurations used in this study are available at this repository.

3.4 Experiment Setup

As described in the previous section, for the experiment setup, two additional versions of each dataset were produced using (Akil et al., 2022) for paraphrasing and (Hasan et al., 2020) for backtransla-

Dataset	Version	mBERT				XLM-Indic				BanglaBert			
		F1-Macro	F1-Micro	Recall	Prec.	F1-Macro	F1-Micro	Recall	Prec.	F1-Macro	F1-Micro	Recall	Prec.
BD-SHS	Baseline	91.9648	**91.9666**	91.9886	92.0494	91.8253	91.8274	91.8441	91.9061	91.9680	91.9666	91.8753	91.8713
	Paraphrased	91.4841	91.4894	91.4733	91.5325	92.3445	**92.3444**	92.4433	92.4692	92.3443	**92.3444**	92.4433	92.3444
	Backtranslated	90.9909	90.9922	90.9922	91.0293	90.5150	90.5150	90.6624	90.5150	91.4295	91.4297	91.6191	91.5874
SentNoB	Baseline	**69.5592**	71.6267	69.5000	69.7422	**68.3493**	70.4288	69.7352	69.7352	69.4012	73.7705	70.3164	69.3870
	Paraphrased	67.3360	69.1677	67.4392	67.6137	67.2832	69.4830	67.3433	67.4474	**72.2954**	75.8512	73.0009	72.1994
	Backtranslated	66.6228	68.1589	67.0720	67.1482	68.2108	70.1135	68.3326	68.5054	72.0372	75.5359	72.6263	71.8748
VITD	Baseline	**65.8050**	70.8333	64.9746	69.2229	66.4206	71.8254	65.1105	68.3673	72.3954	76.8353	71.1721	76.4834
	Paraphrased	65.0452	70.1389	64.1128	69.8489	65.0522	70.6845	65.1723	69.3623	**74.6663**	78.6210	73.7993	77.7827
	Backtranslated	64.1207	69.5933	63.8942	68.2788	**68.1876**	70.2396	68.1758	68.4270	74.3087	78.1746	72.6043	78.9988
BLP Task 2	Baseline	58.6598	**61.0109**	58.6969	60.2731	61.9487	**65.0962**	61.5665	62.8359	66.4386	71.3434	66.2780	66.6474
	Paraphrased	56.4873	59.3112	56.8890	57.4954	59.8386	63.7841	59.53085	60.3757	66.6997	**72.1403**	70.9197	72.1403
	Backtranslated	57.3038	59.9374	57.4055	58.5438	60.4425	63.8288	60.16027	61.2324	64.5253	70.2550	64.9660	64.5774

orange indicates the best performing augmentation method for each model and blue indicates the best performing model within each dataset.

Table 1: Evaluation of models on different datasets and their augmentations

tion. We augmented only the minor class in the training split. It should be noted that the dataset BD-SHS (Romim et al., 2022) is not imbalanced. Nevertheless, we still augmented it to increase its size to twice its original volume.

For our experiments, we used three different models. Here, our main goal was to compare the results between monolingual, language family specific multilingual and general multilingual language models. Hence, we used BanglaBERT (Bhattacharjee et al., 2021), as our monolingual language model, XLM-Indic (Moosa et al., 2023), as our language family specific multilingual language model and finally mBERT (Devlin et al., 2018), as our multilingual model with large number of languages.

All the models were trained for 3 epochs. All the baseline models except XLM-Indic had a learning rate of 2e-5.The learning rate for XLM-Indic was set at 3e-5 for the BLP Task 2 dataset and 4.5e-5 for rest of the baseline datasets. Furthermore, the learning rates required for the augmented datasets were higher and ranged from 3e-5 to 8e-5. Further details on hyperparameters are provided in Appendix A.

3.5 Hyperparameter Rationale

The hyperparameter selection was guided by model and dataset characteristics. All models were trained for 3 epochs, which proved sufficient for convergence without overfitting. Baseline models generally performed well with a learning rate of 2e-5, standard for BERT fine-tuning. However, XLM-Indic required higher learning rates (3e-5 to 4.5e-5) for baseline datasets, likely due to its multilingual pre-training requiring more aggressive updates to adapt to Bangla-specific sentiment patterns.

Augmented datasets consistently required higher learning rates (3e-5 to 8e-5) compared to baselines. This increase was necessary because augmentation introduced greater variance through paraphrasing and backtranslation, requiring models to adapt to a wider distribution of linguistic expressions. The doubled dataset size also necessitated more aggressive gradient steps for convergence within the same epochs. BanglaBERT, being monolingual, showed more stability with moderate learning rate increases, while multilingual models (mBERT and XLM-Indic) needed more varied adjustments to effectively utilize the augmented samples.

4 Results and Discussion

In this study, we explored how generative data augmentation can potentially enhance performance in semantic classification task for Bangla. In the subsequent section, we present the results of our findings and also show how the improvements from augmentations can be affected in presence of noisy labels.

4.1 Data augmentation improves performance

From Table 1, we can see that data augmentation indeed improves the performance of models. However, this improvement is not entirely universal. Here we will discuss from both model and augmentation technique point of view. From augmentation technique side of things, we can observe that as a technique, paraphrasing outperforms backtranslation. For instance, BanglaBert sees an improvement of approximately 2% F1-Micro for BD-SHS, 3% F1-Macro for SentNoB, 3% F1-Micro for VITD and finally 1% F1-Micro on BLP Task 2. Although backtranslation did not show improve-

Dataset	**Classes**	F1-Macro	Accuracy	Recall	Prec.
SentNoB	No Neutral	90.57	90.61	90.58	90.55
	No Positive	79.19	80.79	80.39	78.52
	No Negative	74.12	77.54	76.26	73.14
BLP Task 2	No Neutral	83.88	84.57	83.59	84.28
	No Positive	70.24	76.97	71.06	69.62
	No Negative	75.42	78.24	78.36	74.40
VITD	No Passive Violence	88.77	93.75	86.78	91.23
	No Direct Violence	79.42	81.32	82.35	78.39
	No Non Violence	78.25	82.61	76.14	84.03

Table 2: Impact of label noise on performance

ment in all the datasets for Banglabert, we can still see that it improved the baseline by 2.6% for SentNoB dataset. Apart from that, backtranslation also improved the baseline results of XLM-Indic on VITD by 2% and paraphrase improves BD-SHS baseline by 0.5%. On average paraphrasing gave the best improvements compared to backtranslation. Given the texts are from social media and are noisy by nature, it might be that backtranslation further added some noise due to translation process whereas paraphrasing might have had better noise to variance ratio due to its permutation nature. Another interesting thing to observe is the BD-SHS dataset. Most models performed quite well on it and this is likely due to its classes polar and binary nature. Hence, we did not see as much improvement from augmentation and the results of XLM-Indic and BanglaBert also seems to match for the paraphrased BD-SHS.

4.2 Impact of Pretraining

From a pretraining perspective, we see that the monolingual model performed much better with augmentations compared to the multilingual models. In most cases the multilingual models performed worse than their baseline results with the augmented datasets. This is likely due to the noise to variance ratio we discussed earlier. It seems that, monolingual models are better suited to use the added variance of the augmentation methods compared to multilingual models. This finding is not in line with (Ghosh and Senapati, 2022) where they report multilingual model perform as well as monolingual models on a similar task in Bangla.

4.3 Label Noise Impedes Performance Gains

In Section 4.1, we saw how data augmentation improved performance over baseline. However, this performance could be improved much more with better data quality. For instance, in Figure 3, we showed how these datasets have noisy labels and in some cases really poor inter annotator agreement (Islam et al., 2021) and how these noisy labels may impact performance. Here, we show empirically that, noisy label induced ambiguous decision boundary indeed degrades performance. We show this by performing binary classification on the baseline datasets using BanglaBert. Our main hypothesis here was that, the performance difference between our binary classifiers would not be drastically high if they were represented equally without label noise. However, in presence of label noise, the class boundary would be ambiguous and that would degrade the model. We can exactly see this in Table 2. Here, we can see that on SentNoB, removing the neutral class results in the best performing model. Whereas, in presence of neutral class, we see a 10% reduction of accuracy score. Specially, we can observe that the classifier for positive and neutral classes performed the worst out of all three permutations. It is expected that polarized classes like positive and negative would be easier to learn and neutral classes being somewhat in the middle might be harder to learn. However, here we see considerable degradation. Hence, we believe that the positive class has more overlap with the neutral class and as stated earlier, this results in poorer decision boundaries for classifiers. The baseline result for SentNoB was 69.40 F1-Macro and the best result was paraphrased SentNoB with a score of 72.30 F1-macro. Comparing these to the results in Table 2 can give us some idea how the results degrade due to noisy labels. Furthermore, we can see similar trends for both VITD and BLP Task 2. Both of them show on average almost 10% reduction in accuracy score. As discussed earlier, BLP Task 2 is a mix of SentNoB and MUBASE dataset. Hence, similar to results on 1, it gives us a glimpse of the error propagation of MUBASE. We can see that compared to 90.57 accuracy score of SentNoB's no neutral class subset, the no neutral class subset of BLP Task2 has an accuracy score of 84.57. Again we see that the main issue here is the neutral class. This leads us to believe that neutral class annotation are the main cause of label noise and it requires better attention.

4.4 Issues with Neutral Sentiment

The distinction between neutral and other sentiments is where the model struggles the most, for both positive and negative. Unlike polarized classes, which may have specific lexical indicators, the neutral class lacks such clear markers. We hypothesize that the neutral classes have a much higher distribution than the polarized classes. To accurately represent the distribution of these neutral classes, we recommend a much higher representation of neutral class than the polarized classes.

We would also advocate for the introduction of an "indeterminate" class to address another issue. While a neutral sentiment refers to an unpolarized yet clearly determined sentiment, an indeterminate label captures instances where sentiment is genuinely unclear or ambiguous. Whether this deserves its own category requires further analysis and validation.

By adopting this labeling scheme, we can ensure that the model does not mistakenly categorize uncertain sentiments into the neutral category, thereby preserving the integrity of both classes.

5 Limitations

We focus on encoder-only models to control confounds across monolingual vs. multilingual settings under matched budgets, leaving decoder-only and sequence-to-sequence architectures to future work. While we emphasize sentence-level augmentation for semantic fidelity, a broader comparison with additional augmentation families (e.g., mixup/noising) is also deferred.

6 Conclusion

In this study, we demonstrate that data augmentation techniques, notably paraphrasing and backtranslation, enhance the performance of Bangla sentiment classifiers. Our results reveal that paraphrasing significantly benefits monolingual models, more so than backtranslation does for multilingual models. We also detected noisy labels across all four datasets. Our analysis provides empirical evidence that label noise hampers classifier performance, with the neutral class emerging as the primary source of this noise. Given these findings, we argue for robust protocols for annotating neutral classes. We propose weighting inter-annotator agreement by class, suggesting the neutral class be assigned the highest weight. Consequently, the neutral class should attain higher inter-annotator agreement scores compared to the positive and negative classes.

References

S. M. Abu Taher, Kazi Afsana Akhter, and K. M. Azharul Hasan. 2018. N-gram based sentiment mining for Bangla text using support vector machine. In *Proceedings of the 2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5.

Ajwad Akil, Najrin Sultana, Abhik Bhattacharjee, and Rifat Shahriyar. 2022. Banglaparaphrase: A high-quality bangla paraphrase dataset. *arXiv preprint arXiv:2210.05109*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

Anirban Bhowmick and Abhik Jana. 2021. Sentiment analysis for Bengali using transformer based models. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 481–486, National Institute of Technology Silchar, Silchar, India. NLP Association of India (NLPAI).

Nitish Ranjan Bhowmik, Mohammad Arifuzzaman, and M. Rubaiyat Hossain Mondal. 2022. Sentiment analysis on bangla text using extended lexicon dictionary and deep learning algorithms. *Array*, 13:100123.

Partha Chakraborty, Farah Nawar, and Humayra Afrin Chowdhury. 2022. A ternary sentiment classification of Bangla text data using support vector machine and random forest classifier. In Jyotsna Kumar Mandal, Pao-Ann Hsiung, and Rudra Sankar

Dhar, editors, *Topical Drifts in Intelligent Computing*, pages 69–77. Springer Nature Singapore, Singapore.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Koyel Ghosh and Dr. Apurbalal Senapati. 2022. Hate speech detection: a comparison of mono and multilingual transformer model with cross-language evaluation. In *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation*, pages 853–865, Manila, Philippines. De La Salle University.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for bengali-english machine translation. *arXiv preprint arXiv:2009.09359*.

Eftekhar Hossain, Omar Sharif, Mohammed Moshiul Hoque, and Iqbal H Sarker. 2020. Sentilstm: a deep learning approach for sentiment analysis of restaurant reviews. In *International Conference on Hybrid Intelligent Systems*, pages 193–203. Springer.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Khondoker Ittehadul Islam, Md Saiful Islam, and Md Ruhul Amin. 2020. Sentiment analysis in bengali via transfer learning using multi-lingual bert. In *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, pages 1–5. IEEE.

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.

Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. 2020. Learning the difference that makes a difference with counterfactually augmented data. In *International Conference on Learning Representations*.

Mst Eshita Khatun and Tapasy Rabeya. 2022. A machine learning approach for sentiment analysis of book reviews in bangla language. In *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1178–1182. IEEE.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2011. Model-portability experiments for textual temporal analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 271–276, Portland, Oregon, USA. Association for Computational Linguistics.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Ibraheem Muhammad Moosa, Mahmud Elahi Akhter, and Ashfia Binte Habib. 2023. Does transliteration help multilingual language modeling? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 670–685, Dubrovnik, Croatia. Association for Computational Linguistics.

Forhad An Naim. 2021. Bangla aspect-based sentiment analysis based on corresponding term extraction. In *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, pages 65–69.

Md Rafi-Ur-Rashid, Mahim Mahbub, and Muhammad Abdullah Adnan. 2022. Breaking the curse of class imbalance: Bangla text classification. *Transactions on Asian and Low-Resource Language Information Processing*, 21(5):1–21.

Shadman Rohan, Mojammel Hossain, Mohammad Mamun Or Rashid, and Nabeel Mohammed. 2023. Bencoref: A multi-domain dataset of nominal phrases and pronominal reference annotations. In *Proceedings of the 17th Linguistic Annotation Workshop (LAW)*.

Nauros Romim, Mosahed Ahmed, Md Saiful Islam, Arnab Sen Sharma, Hriteshwar Talukder, and Mohammad Ruhul Amin. 2022. BD-SHS: A benchmark dataset for learning to detect online Bangla hate speech in different social contexts. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5153–5162, Marseille, France. European Language Resources Association.

Sourav Saha, Jahedul Alam Junaed, Maryam Saleki, Mohamed Rahouti, Nabeel Mohammed, and Mohammad Ruhul Amin. 2023. Blp-2023 task 1: Violence inciting text detection (vitd). In *Proceedings of the 1st International Workshop on Bangla Language Processing (BLP-2023)*, Singapore. Association for Computational Linguistics.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Md Ferdous Wahid, Md Jahid Hasan, and Md Shahin Alom. 2019. Cricket sentiment analysis from bangla text using recurrent neural network with long short term memory model. In *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–4. IEEE.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33. Curran Associates, Inc.

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.

A Finetuning Hyperparameters

Dataset	Batch Size	Learning Rate	Weight Decay	Dropout	Epochs	Warmup Ratio	Label Smoothing
BD-SHS Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
BD-SHS Paraphrased	32	5e-5	1e-6	0.15	3	0.10	0.15
BD-SHS Backtranslated	32	6e-5	1e-3	0.15	3	0.10	0.15
SentNoB Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
SentNoB Paraphrased	32	5.5e-5	1e-3	0.15	3	0.10	0.15
SentNoB Backtranslated	32	2e-5	1e-3	0.15	3	0.10	0.15
VITD Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
VITD Paraphrased	32	3e-5	1e-3	0.15	3	0.10	0.15
VITD Backtranslated	32	6e-5	1e-3	0.15	3	0.10	0.15
BLP Task 2 Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
BLP Task 2 Paraphrased	32	3e-5	1e-6	0.15	3	0.10	0.15
BLP Task 2 Backtranslated	32	6e-5	1e-3	0.15	3	0.10	0.15

Table 3: Hyperparameters for mBERT model

Dataset	Batch Size	Learning Rate	Weight Decay	Dropout	Epochs	Warmup Ratio	Label Smoothing
BD-SHS Baseline	32	4.5e-5	1e-6	0.15	3	0.10	0.15
BD-SHS Paraphrased	32	5e-5	1e-3	0.15	3	0.10	0.15
BD-SHS Backtranslated	32	6e-5	1e-6	0.15	3	0.10	0.15
SentNoB Baseline	32	4.5e-5	1e-6	0.15	3	0.10	0.15
SentNoB Paraphrased	32	6e-5	1e-6	0.15	3	0.10	0.15
SentNoB Backtranslated	32	3.5e-5	1e-6	0.15	3	0.10	0.15
VITD Baseline	32	4.5e-5	1e-6	0.15	3	0.10	0.15
VITD Paraphrased	32	5.5e-5	1e-3	0.15	3	0.10	0.15
VITD Backtranslated	32	4e-5	1e-6	0.15	3	0.10	0.15
BLP Task 2 Baseline	32	3e-5	1e-6	0.15	3	0.10	0.15
BLP Task 2 Paraphrased	32	4e-5	1e-6	0.15	3	0.10	0.15
BLP Task 2 Backtranslated	32	3e-5	1e-6	0.15	3	0.10	0.15

Table 4: Hyperparameters for XLM-Indic model

Dataset	Batch Size	Learning Rate	Weight Decay	Dropout	Epochs	Warmup Ratio	Label Smoothing
BD-SHS Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
BD-SHS Paraphrased	32	5e-5	1e-3	0.15	3	0.10	0.15
BD-SHS Backtranslated	32	5e-5	1e-3	0.15	3	0.10	0.15
SentNoB Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
SentNoB Paraphrased	32	5e-5	1e-3	0.15	3	0.10	0.15
SentNoB Backtranslated	32	8e-5	1e-3	0.15	3	0.10	0.15
VITD Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
VITD Paraphrased	32	5e-5	1e-3	0.15	3	0.10	0.15
VITD Backtranslated	32	4e-5	1e-3	0.15	3	0.10	0.15
BLP Task 2 Baseline	32	2e-5	1e-3	0.15	3	0.10	0.15
BLP Task 2 Paraphrased	32	2e-5	1e-3	0.15	3	0.10	0.15
BLP Task 2 Backtranslated	32	5e-5	1e-3	0.15	3	0.10	0.15

Table 5: Hyperparameters for BanglaBERT model

bnContextQA: Benchmarking Long-Context Question Answering and Challenges in Bangla

Adnan Ahmad[1], Labiba Adiba[1], Namirah Rasul[1]*
Md Tahmid Rahman Laskar[2], Sabbir Ahmed[1]
[1]Islamic University of Technology, [2]York University
[1]{adnanahmad, labibaadiba, namirahrasul, sabbirahmed}@iut-dhaka.edu
[2]tahmid20@yorku.ca

Abstract

Large models have advanced in processing long input sequences, but their ability to consistently use information across extended contexts remains a challenge. Recent studies highlight a positional bias where models prioritize information at the beginning or end of the input while neglecting the middle, resulting in a U-shaped performance curve, but this was limited to English. Whether this bias is universal or shaped by language-specific factors remains unclear. In this work, we investigate positional bias in Bangla, a widely spoken but computationally underrepresented language. To support this, we introduce a novel Bangla benchmark dataset, 'bnContextQA', specifically designed for long-context comprehension. The dataset comprises of 350 long-context QA instances, each paired with 30 context paragraphs, allowing controlled evaluation of information retrieval at different positions. Using this dataset, we assess the performance of LLMs on Bangla across varying passage positions, providing insights into cross-linguistic positional effects. The bnContextQA dataset is publicly available at https://github.com/labiba02/bnContextQA.git to support future research on long-context understanding in Bangla and multilingual LLMs.

1 Introduction

Large language models are increasingly capable of processing long sequences, with context lengths extending to tens of thousands of tokens (Chang et al., 2024). This capability is crucial for real-world applications, including question answering, summarization, and retrieval-augmented generation (Zheng et al., 2025; Laskar et al., 2023, 2024). However, the assumption that models can robustly use all available context is being challenged by recent findings (Li et al., 2024a).

The Lost in the Middle study(Liu et al., 2023) highlights a striking limitation: LLMs tend to prioritize information located at the beginning (primacy bias) or end (recency bias) of their input, while struggling to retrieve and apply information positioned in the middle. This U-shaped performance curve calls into question the practical utility of extended context lengths, since critical details in real documents are not always conveniently placed.

Despite these findings, prior research has been restricted to English. Given the syntactic and morphological differences in languages like Bangla, it is vital to investigate if the same biases persist across multilingual contexts.

Bangla, being one of the most widely spoken languages in the world, remains underexplored in the evaluation of LLMs (Kabir et al., 2024; Mahfuz et al., 2025; Abrar et al., 2024). Unlike English, Bangla has complex morphology and flexible word order, which may interact differently with model architectures when processing long contexts. In this work, we address this gap by:

1. Constructing a Bangla long-context QA dataset with 350 questions and 30 passages per question.
2. Running baseline evaluations with two state-of-the-art generative QA models, GPT-4.1 (OpenAI) and Gemini 2.5 Flash Lite (Google)
3. Presenting early evidence of positional bias in Bangla.

Our preliminary results show that, similar to English, LLMs also struggle with middle-position evidence in Bangla. These findings motivate further work on long-context modeling, dataset expansion, and evaluation of Bangla LLMs.

2 Literature Review

Recent progress in the advanced language models has excelled across a broad spectrum of natural lan-

[*] Authors 1,2,3 contributed equally.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 357–365
December 23, 2025 ©2025 Association for Computational Linguistics

guage tasks, enabling them to adapt to diverse linguistic settings (Mahbub et al., 2023; Ahmed et al., 2024; Khan et al., 2023a,b; Arif et al., 2025). As their capabilities expand, researchers have increasingly turned their attention to how such models perform in long-context processing, where models must maintain coherence, track dispersed information, and reliably retrieve details embedded across lengthy sequences (Li et al., 2024a; Huang et al., 2024; Liu et al., 2025).

Prior works on long-context processing have primarily focused on English (Bai et al., 2024; Li et al., 2024b; Bai et al., 2025; Gao et al., 2025). A key study, Lost in the Middle: How Language Models Use Long Contexts (Liu et al., 2023), systematically examined model behavior on extended inputs, showing that accuracy drops for middle-position evidence. However, it relied solely on English datasets, leaving open whether these positional effects generalize to other languages.

Liu et al. (2023) evaluated positional sensitivity using two tasks:

- **Multi-document question answering**: Models received multiple documents with only one containing the correct answer, using the NaturalQuestions-Open (NQ-Open) dataset (Lee et al., 2019). The gold document's position was varied to measure robustness.
- **Key–value retrieval**: A synthetic benchmark where models selected the correct value from a JSON object, allowing position to be manipulated independently of natural language semantics.

Across both settings, performance followed a U-shaped curve, with strong primacy and recency effects and significantly weaker retrieval in the middle.

While these findings demonstrate positional bias, existing resources offer little insight into how this manifests in multilingual or low-resource settings. In Bangla, current reading-comprehension datasets rely on single-document passages and therefore cannot evaluate long-context reasoning or positional effects. The most notable dataset, BanglaRQA (Ekram et al., 2022), provides 3000 passages and 14,889 question–answer pairs with diverse question types and answer formats, but lacks multi-document inputs, distractors, and controlled evidence placement. This gap motivates the creation of bnContextQA, a benchmark designed specifically for long-context comprehension in Bangla. By pairing each question with 30 semantically related passages, including curated distractors, and precisely controlling the gold passage's position, our dataset enables systematic analysis of positional bias and extends long-context QA research beyond English.

3 bnContextQA

In this section, we explain the construction of our long-context Bangla QA dataset. We describe how passages were collected and curated from Bangla Wikipedia, how distractor passages were carefully designed to be topically similar yet unanswerable, and the preprocessing steps applied to ensure data quality. An example of a dataset instance and summary statistics are provided in Appendix A.1.

3.1 Data Acquisition

To study the effect of long-context input on Bangla LLMs, we constructed a Bangla long-context QA dataset simulating multi-document question answering. Existing Bangla QA datasets, such as Bengali-SQuAD (Tahsin Mayeesha et al., 2021), SQuAD_Bn (Bhattacharjee et al., 2022), and BanglaQA (Shahriar et al., 2023), mainly contain short passages, limiting systematic evaluation on extended contexts. Our dataset includes multiple passages per question, with one gold passage containing the answer and several semantically related distractors.

We used Bangla Wikipedia as the primary source for its broad coverage across domains. Passages were manually curated to ensure correctness, quality, and domain diversity, and to control semantic similarity for realistic distractors—beyond what automatic extraction can reliably achieve. This careful construction ensures the QA task requires genuine reasoning rather than superficial keyword matching, providing a robust benchmark for long-context comprehension in LLMs.

3.2 Dataset Structure

Each sample in our dataset is represented as a JSON object with the following components:

- **Question** (`question`): A natural language query in Bangla.
- **Language** (`language`): Fixed as ”bn” to indicate Bengali.
- **Documents** (`documents`): A list of 30 passages, each containing a title, content, and

source. One passage contains the gold evidence for the correct answer. The remaining passages serve as distractors, deliberately chosen to share topical or lexical similarity with the gold passage, increasing task difficulty. Each passage is of 175 token on average.

- **Answer** (`answer`): The ground-truth answer derived from the relevant document.
- **Relevant Document Index**(`relevant_document_index`): The index pointing to the passage containing the gold evidence.
- **Context Length** (`context_length`): The number of passages provided per instance (fixed at 30 in our dataset).
- **Metadata** (`metadata`): Includes dataset source, retrieval method, and special notes.

3.3 Question Generation

To create queries for multi-document evaluation, we selected topics from Banglapedia and Wikibangla that contain many closely related articles, ensuring sufficient semantic overlap between gold passages and distractors. Using these topics as prompts, we generated candidate questions with ChatGPT and then manually filtered them to ensure short, unambiguous answers that were not easily guessable from the gold passage's wording or position. Throughout the process, prompts were crafted carefully so the model could not rely on prior knowledge but had to use the provided gold passage, ensuring alignment with the objectives of evaluating long-context reasoning.

3.4 Distractor Design

A key feature of our dataset is the careful construction of distractor passages. Instead of random text, distractors were selected from Bangla Wikipedia and Banglapedia to ensure topical and stylistic alignment with the gold passage.

To keep this process systematic, native Bangla speakers applied a structured manual filtering procedure. Candidate distractors were evaluated according to:

- **Topical relevance:** The distractor must fall within the same broad domain as the gold passage to maintain thematic coherence (e.g., historical sites, political events, scientific topics).
- **Lexical similarity:** Passages were selected to share key vocabulary, technical terms, or stylistic features with the gold passage, preventing models from relying on simple keyword matching.
- **Factual distinction:** Distractors were checked to ensure they contain no answer-bearing text or paraphrases that could accidentally reveal the correct answer.
- **Structural parity:** Distractors were matched to the gold passage in length, complexity, and informational density, preventing models from exploiting superficial cues such as unusually short, long, or structurally simple passages.

This human-guided design produces distractors that are plausible, challenging, and semantically aligned, resulting in a robust evaluation setting for long-context comprehension in Bangla LLMs.

3.5 Preprocessing and Cleaning

During passage collection, raw Wikipedia text often contained nuanced references, extraneous symbols, or English phrases that could bias results. To minimize such noise, annotators were instructed to remove redundant citations and bracketed references, normalize the Bangla script to a consistent Unicode form, eliminate repetitive English words unless essential to factual content (translating necessary terms into Bangla), and standardize passage lengths to ensure comparability. Additional metadata such as topic category, article identifiers, and the specified gold position was added to each item. This produced a dataset that is linguistically coherent, semantically consistent, and suitable for evaluating long-context reasoning in Bangla.

4 Experimental Details

4.1 Models

In this section, we describe the models that we evaluate on our proposed Bangla dataset. We conducted our experiments using Gemini-2.5-Flash-Lite (Google, 2024) and GPT-4.1-Nano (OpenAI, 2024) (details about the models are given in Appendix A.2). We selected them for their cost-effectiveness and accessibility. Gemini-2.5-Flash-Lite offers high-throughput processing at low cost while maintaining strong reasoning capabilities. GPT-4.1-Nano provides efficient performance with extensive context windows. These attributes make both models particularly suitable for deployment in Bangla-speaking regions where computational resources are often constrained. By focusing on these models, our work evaluates whether affordable LLMs can still maintain competitive performance on long-context tasks in low-resource languages.

Although large generative models like TituLLM (Nahin et al., 2025) and BongLLaMA (Zehady et al., 2024) are trained on Bangla language and support extended contexts, we did not use them in this study. In our preliminary trials, we observed that these models tended to overgenerate or rely heavily on parametric knowledge rather than grounding their answers in the provided passages. Since our evaluation requires short, span-based answers to measure sensitivity to positional placement, such behavior makes them less suitable for this specific task.

4.2 Evaluation Method

Each model was evaluated across all context lengths and gold passage positions. Models predicted the answer span from one gold passage and multiple semantically related distractors, increasing task difficulty and requiring fine-grained reasoning.

To assess positional effects, the gold passage was placed at multiple locations across different context lengths: positions 1, 3, and 5 for length 5; 1, 3, 5, 7, and 10 for length 10; 1, 5, 10, 15, and 20 for length 20; and 1, 5, 10, 15, 20, 25, and 30 for length 30. These placements capture early, middle, and late positions, enabling analysis of primacy and recency effects as well as mid-sequence degradation. Combined with controlled context variation and carefully matched distractors, this setup provides a robust framework for evaluating Bangla QA models' long-context reasoning and extractive accuracy. Details on evaluation metrics and implementation are provided in Appendices A.3 and A.4.

5 Results

We evaluated two generative question answering models, `GPT-4.1-nano` and `Gemini 2.5-flash-lite` on our Bangla QA dataset with input contexts containing 10, 20, and 30 total passages, each containing one gold passage and several distractors.

The results in Table 1 demonstrate a clear positional effect on model performance for both GPT-4.1-nano and Gemini-2.5-Flash-Lite. For GPT-4.1-nano, the highest scores are observed when the relevant context appears at the beginning (Index 1), with 60.63% Exact Match (EM) and 67.40% F1, followed by a sharp decline in the middle positions (Indices 3 and 5) and a slight recovery at Index 10 (41.59% EM, 48.00% F1). Gemini-2.5-Flash-Lite shows a similar U-shaped trend but consistently achieves higher overall scores, with EM and F1 peaking at 61.59% and 76.32% at Index 1, gradually decreasing toward the middle, and partially recovering at Index 10 (54.29% EM, 71.30% F1). This pattern indicates that both models pay more attention to information at the beginning and end of long contexts while underperforming for passages in the middle, confirming that positional bias also manifests in Bangla QA tasks and mirrors the "lost in the middle" phenomenon observed in prior studies (Liu et al., 2023).

Table 1: Evaluation results of GPT-4.1-nano and Gemini-2.5-flash-lite on different indices, showing Exact Match (EM) and F1 Score.

Index	**GPT-4.1-nano**		**Gemini-2.5-Flash-Lite**	
	EM	F1	EM	F1
1	60.63%	67.40%	61.59%	76.32%
3	51.11%	57.36%	57.78%	73.13%
5	40.95%	48.23%	56.19%	72.29%
7	38.73%	43.86%	56.83%	72.16%
10	41.59%	48.00%	54.29%	71.30%

Similar positional patterns emerge across all context lengths for both models. For length 5, GPT-4.1-nano performs best at early positions (64.13% EM, 69.42% F1) but drops in the middle, while Gemini-2.5-Flash-Lite maintains higher and more stable scores (60.63–60.95% EM, 74.50–75.66% F1) with only a mild mid-sequence dip. For longer contexts (20 and 30), the U-shaped trend becomes more pronounced: GPT-4.1-nano starts relatively high but declines sharply in the middle (down to 34.92% EM, 40.27% F1 at length 20 and 30.16% EM, 34.09% F1 at length 30) before a slight end-of-sequence recovery. Gemini-2.5-Flash-Lite follows the same pattern but consistently outperforms GPT-4.1-nano, maintaining stronger EM/F1 scores even at mid-range positions (e.g., 54.60–55.56% EM and 64.49–70.93% F1). Overall, this confirms a robust U-shaped positional bias in Bangla long-context QA, consistent with findings from English benchmarks.

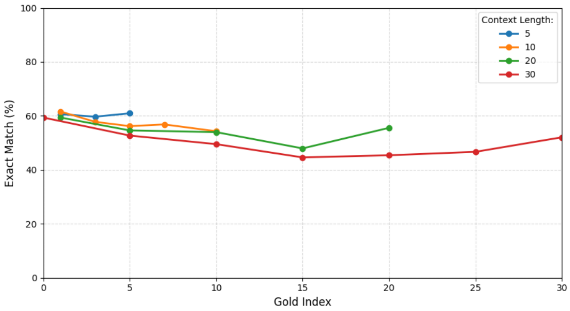


Figure 1: Performance comparison using Exact Match in Gemini-2.5-Flash-Lite

Figure 1 shows the performance of Gemini-2.5-Flash-Lite on different context lengths and gold passage positions using Exact Match. Detailed results for context lengths 5, 20, and 30 are provided in Appendix Tables 3, 4, and 5 along with performance graph of the models for visualization in Figure 3- 5.

Why Gemini Outperforms GPT-4.1-Nano: While both models show a similar U-shaped positional pattern, Gemini-2.5-Flash-Lite consistently achieves higher EM and F1 scores across all context lengths. Several factors may explain this gap. First, architectural differences may give Gemini stronger long-context retrieval, such as improved attention routing or memory-efficient mechanisms. Second, its tokenizer offers better subword coverage for Indo-Aryan languages, reducing Bangla word fragmentation and improving span extraction. Third, Gemini's broader multilingual and South Asian training corpus likely provides richer exposure to Bangla morphology, orthographic variation, and Wikipedia-style text. Together, these advantages help Gemini maintain more reliable attention over long Bangla contexts, especially in the middle regions where GPT-4.1-Nano degrades more sharply.

6 Conclusion

In this work, we introduced a Bangla long-context QA dataset with semantically challenging distractors and reported preliminary results on generative QA models, GPT-4.1-Nano and Gemini-2.5-Flash-Lite. Our early findings confirm positional biases similar to the "Lost in the Middle" phenomenon observed in English: models achieve higher accuracy when the gold passage appears at the beginning or end of the context, but struggle when it is placed in the middle. Among the tested models, Gemini-2.5-Flash-Lite consistently outperformed GPT-4.1-nano. These results represent an initial step rather than a complete solution. Our ongoing work focuses on two directions: (1) evaluating more Bangla LLMs while enforcing grounding in provided passages (like TigerLLM (Raihan and Zampieri, 2025)), and (2) expanding the dataset with more questions, diverse domains, and multi-hop reasoning. By pursuing these directions, we aim to provide a stronger benchmark and a more comprehensive understanding of how Bangla LLMs process extended contexts. Other task, like intrinsic bias measurements of Bangla (Sadhu et al., 2024), can be done for longer context and context length variation using our dataset.

Limitations

Despite providing a first step toward long-context QA in Bangla, our study has several limitations. The dataset is small, with each instance containing only a single gold passage, limiting multi-hop reasoning and domain coverage. Distractor passages, though carefully designed, may not fully capture real-world complexity, and evaluation of a few LLMs using span-based metrics (Exact Match and F1) may not reflect generative answer quality. Moreover, we did not focus on Bangla-specific linguistic aspects such as morphology and syntax, which remain important directions for future work. Overall, these results are preliminary, and further work is needed to expand the dataset, explore more reasoning scenarios, and test additional models.

Acknowledgments

The authors would like to express their sincere gratitude to the Islamic University of Technology (IUT) for providing financial support during the dataset curation phase. The authors also acknowledge the CUPE Research Grant Fund for supporting the OpenAI API model usage costs.

References

Ajwad Abrar, Farzana Tabassum, and Sabbir Ahmed. 2024. Performance evaluation of large language models in bangla consumer health query summarization. In *2024 27th International Conference on Computer and Information Technology (ICCIT)*, pages 2748–2753.

Tasnim Ahmed, Shahriar Ivan, Ahnaf Munir, and Sabbir Ahmed. 2024. Decoding depression: Analyzing social network insights for depression severity assessment with transformers and explainable ai. *Natural Language Processing Journal*, 7:100079.

Nokimul Hasan Arif, Shadman Rabby, Md Hefzul Hossain Papon, and Sabbir Ahmed. 2025. Preemptive hallucination reduction: An input-level approach for multimodal language model. *Preprint*, arXiv:2505.24007.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, Vienna, Austria. Association for Computational Linguistics.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, and Rifat Shahriyar. 2022. Banglanlg: Benchmarks and resources for evaluating low-resource natural language generation in bangla. *CoRR, abs/2205.11081*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3).

Syed Mohammed Sartaj Ekram, Adham Arik Rahman, Md. Sajid Altaf, Mohammed Saidul Islam, Mehrab Mustafy Rahman, Md Mezbaur Rahman, Md Azam Hossain, and Abu Raihan Mostofa Kamal. 2022. BanglaRQA: A benchmark dataset for under-resourced Bangla language reading comprehension-based question answering with diverse question-answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2518–2532, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. 2025. How to train long-context language models (effectively). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7376–7399, Vienna, Austria. Association for Computational Linguistics.

Google. 2024. Gemini 2.5 flash: Our next-generation model for efficiency. `https://deepmind.google/technologies/gemini/pro/`. Accessed: 2024-11-19.

Yunpeng Huang, Jingwei Xu, Junyu Lai, Zixu Jiang, Taolue Chen, Zenan Li, Yuan Yao, Xiaoxing Ma, Lijuan Yang, Hao Chen, Shupeng Li, and Penghao Zhao. 2024. Advancing transformer architecture in long-context large language models: A comprehensive survey. *Preprint*, arXiv:2311.12351.

Mohsinul Kabir, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Mir Tafseer Nayeem, M Saiful Bari, and Enamul Hoque. 2024. BenLLM-eval: A comprehensive evaluation into the potentials and pitfalls of large language models on Bengali NLP. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2238–2252, Torino, Italia. ELRA and ICCL.

Alvi Khan, Fida Kamal, Mohammad Abrar Chowdhury, Tasnim Ahmed, Md Tahmid Rahman Laskar, and Sabbir Ahmed. 2023a. BanglaCHQ-summ: An abstractive summarization dataset for medical queries in Bangla conversational speech. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 85–93, Singapore. Association for Computational Linguistics.

Alvi Khan, Fida Kamal, Nuzhat Nower, Tasnim Ahmed, Sabbir Ahmed, and Tareque Chowdhury. 2023b. NERvous about my health: Constructing a Bengali medical named entity recognition dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5768–5774, Singapore. Association for Computational Linguistics.

Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, Enamul Hoque, Shafiq Joty, and Jimmy Huang. 2024. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13785–13816, Miami, Florida, USA. Association for Computational Linguistics.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Huang. 2023. A systematic study and comprehensive evaluation of ChatGPT on benchmark datasets. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 431–469, Toronto, Canada. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2024a. LooGLE: Can long-context language models understand long contexts? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16304–16333, Bangkok, Thailand. Association for Computational Linguistics.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024b. Long-context llms struggle with long in-context learning. *Preprint*, arXiv:2404.02060.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, and 18 others. 2025. A comprehensive survey on long context language modeling. *Preprint*, arXiv:2503.17407.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Ridwan Mahbub, Ifrad Khan, Samiha Anuva, Md Shihab Shahriar, Md Tahmid Rahman Laskar, and Sabbir Ahmed. 2023. Unveiling the essence of poetry: Introducing a comprehensive dataset and benchmark for poem summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14878–14886, Singapore. Association for Computational Linguistics.

Tamzeed Mahfuz, Satak Kumar Dey, Ruwad Naswan, Hasnaen Adil, Khondker Salman Sayeed, and Haz Sameen Shahgir. 2025. Too late to train, too early to use? a study on necessity and viability of low-resource Bengali LLMs. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 1183–1200, Abu Dhabi, UAE. Association for Computational Linguistics.

Shahriar Kabir Nahin, Rabindra Nath Nandi, Sagor Sarker, Quazi Sarwar Muhtaseem, Md Kowsher, Apu Chandraw Shill, Md Ibrahim, Mehadi Hasan Menon, Tareq Al Muntasir, and Firoj Alam. 2025. Titullms: A family of bangla llms with comprehensive benchmarking. *arXiv preprint arXiv:2502.11187*.

OpenAI. 2024. Gpt-4.1 series: Technical report. `https://openai.com`. Accessed: 2024-11-19.

Nishat Raihan and Marcos Zampieri. 2025. Tigerllm- a family of bangla large language models. *arXiv preprint arXiv:2503.10995*.

Jayanta Sadhu, Ayan Khan, Abhik Bhattacharjee, and Rifat Shahriyar. 2024. An empirical study on the characteristics of bias upon context length variation for bangla. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1501–1520.

Md Shihab Shahriar, Ahmad Al Fayad Chowdhury, Md Amimul Ehsan, and Abu Raihan Kamal. 2023. Question answer generation in bengali: Mitigating the scarcity of qa datasets in a low-resource language. 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 430–441.

Tasmiah Tahsin Mayeesha, Abdullah Md Sarwar, and Rashedur M Rahman. 2021. Deep learning based question answering system in bengali. *Journal of Information and Telecommunication*, 5(2):145–178.

Abdullah Khan Zehady, Safi Al Mamun, Naymul Islam, and Santu Karmaker. 2024. Bongllama: Llama for bangla language. *arXiv preprint arXiv:2410.21200*.

Xu Zheng, Ziqiao Weng, Yuanhuiyi Lyu, Lutao Jiang, Haiwei Xue, Bin Ren, Danda Paudel, Nicu Sebe, Luc Van Gool, and Xuming Hu. 2025. Retrieval augmented generation and understanding in vision: A survey and new outlook. *Preprint*, arXiv:2503.18016.

A Appendix

A.1 Dataset Example and Statistics

Each question in our dataset is stored in JSON format, containing the question, a list of passages (with title, content, and source), the gold answer, and metadata. Figure 2 shows a sample instance (truncated for space).

To provide an overview of our dataset, Table 2 summarizes the key statistics, including the total number of items and passages, as well as the average passage length in terms of tokens and characters.

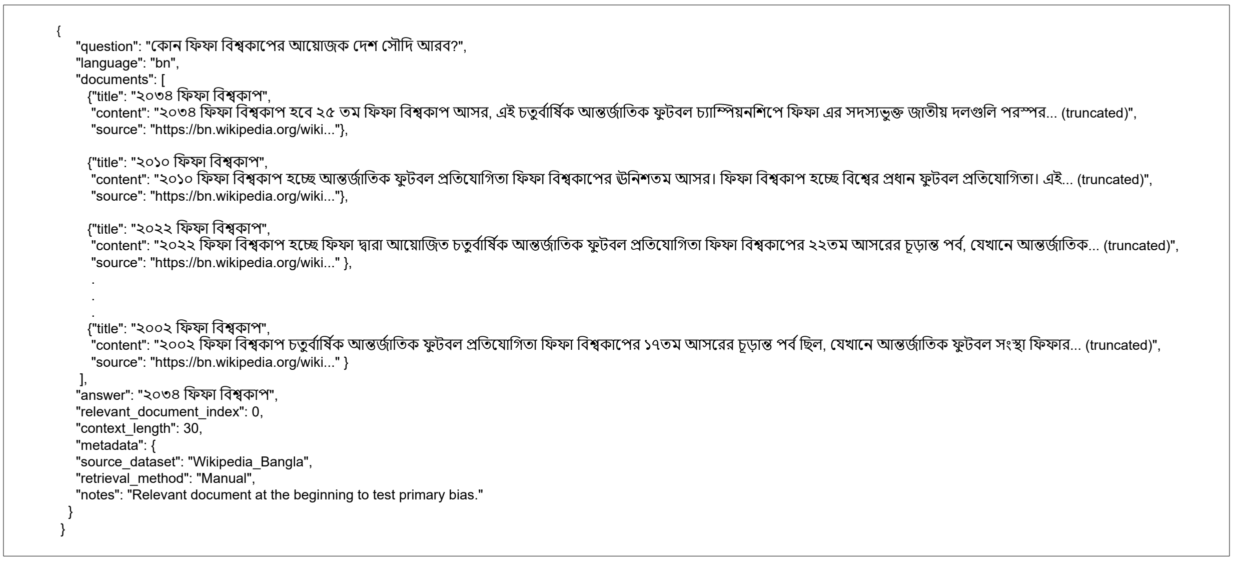
Table 2: Summary statistics of the dataset

Statistic	**Value**
Total items	350
Total passages	10,500
Avg. passage length (tokens)	175.62
Avg. passage length (characters)	1,130.41

A.2 Model Details

We evaluated two extractive QA models on our dataset.

Gemini-2.5-Flash-Lite: Gemini-2.5-Flash-Lite (Google, 2024) is part of Google's Gemini family of models, designed specifically for low latency, high throughput, and cost efficiency. Despite being a lightweight model, it supports up to 1 million tokens of context, making it



```
{
  "question": "কোন ফিফা বিশ্বকাপের আয়োজক দেশ সৌদি আরব?",
  "language": "bn",
  "documents": [
    {"title": "২০৩৪ ফিফা বিশ্বকাপ",
     "content": "২০৩৪ ফিফা বিশ্বকাপ হবে ২৫ তম ফিফা বিশ্বকাপ আসর, এই চতুর্বার্ষিক আন্তর্জাতিক ফুটবল চ্যাম্পিয়নশিপে ফিফা এর সদস্যভুক্ত জাতীয় দলগুলি পরস্পর... (truncated)",
     "source": "https://bn.wikipedia.org/wiki..."},

    {"title": "২০১০ ফিফা বিশ্বকাপ",
     "content": "২০১০ ফিফা বিশ্বকাপ হচ্ছে আন্তর্জাতিক ফুটবল প্রতিযোগিতা ফিফা বিশ্বকাপের ঊনিশতম আসর। ফিফা বিশ্বকাপ হচ্ছে বিশ্বের প্রধান ফুটবল প্রতিযোগিতা। এই... (truncated)",
     "source": "https://bn.wikipedia.org/wiki..."},

    {"title": "২০২২ ফিফা বিশ্বকাপ",
     "content": "২০২২ ফিফা বিশ্বকাপ হচ্ছে ফিফা দ্বারা আয়োজিত চতুর্বার্ষিক আন্তর্জাতিক ফুটবল প্রতিযোগিতা ফিফা বিশ্বকাপের ২২তম আসরের চূড়ান্ত পর্ব, যেখানে আন্তর্জাতিক... (truncated)",
     "source": "https://bn.wikipedia.org/wiki..." },
     .
     .
     .
    {"title": "২০০২ ফিফা বিশ্বকাপ",
     "content": "২০০২ ফিফা বিশ্বকাপ চতুর্বার্ষিক আন্তর্জাতিক ফুটবল প্রতিযোগিতা ফিফা বিশ্বকাপের ১৭তম আসরের চূড়ান্ত পর্ব ছিল, যেখানে আন্তর্জাতিক ফুটবল সংস্থা ফিফার... (truncated)",
     "source": "https://bn.wikipedia.org/wiki..." }
   ],
  "answer": "২০৩৪ ফিফা বিশ্বকাপ",
  "relevant_document_index": 0,
  "context_length": 30,
  "metadata": {
  "source_dataset": "Wikipedia_Bangla",
  "retrieval_method": "Manual",
  "notes": "Relevant document at the beginning to test primary bias."
 }
}
```

Figure 2: Example of an instance from our dataset in JSON format.

suitable for long-context tasks while maintaining affordability. Its release emphasizes stability and accessibility, with one of the lowest per-token costs among commercial LLMs, making it an appealing choice for researchers and practitioners working in resource-constrained environments.

GPT-4.1-Nano: GPT-4.1-Nano (OpenAI, 2024) is the smallest and most affordable member of OpenAI's GPT-4.1 family, introduced in 2025. Like Gemini-2.5-Flash-Lite, it supports a 1 million token context window, enabling it to handle extended inputs effectively. GPT-4.1-Nano is marketed as the fastest and cheapest variant in the GPT-4.1 lineup, optimized for deployment in cost-sensitive or large-scale applications. Despite its reduced size, it demonstrates strong reasoning and comprehension capabilities, striking a balance between performance and accessibility.

A.3 Evaluation Metrics

We evaluate QA model performance using two widely adopted metrics: Exact Match (EM) and F1 score. These metrics provide complementary perspectives on model accuracy.

Exact Match (EM): Exact Match measures the percentage of predictions that exactly match the reference answers. It is a strict metric: a prediction is counted as correct only if it exactly matches the gold answer after normalizing for punctuation, articles, and capitalization. EM is particularly useful for assessing models in scenarios where precise answers are required, such as extractive QA tasks. However, it does not reward partially correct answers or alternative phrasings, making it less informative for generative models that may produce valid but slightly different answers.

F1 Score: The F1 score captures the token-level overlap between the predicted and reference answers, allowing partial credit for answers that are mostly correct. It is the harmonic mean of precision and recall, defined as follows:

$$\text{Precision} = \frac{|\text{pred tokens} \cap \text{ref tokens}|}{|\text{pred tokens}|}$$

$$\text{Recall} = \frac{|\text{pred tokens} \cap \text{ref tokens}|}{|\text{ref tokens}|}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision measures the proportion of predicted tokens that are correct, while recall measures the proportion of reference tokens that are captured by the prediction. F1 balances both aspects, providing a finer-grained evaluation. This metric is especially suitable for generative QA models, which may produce answers that are semantically correct but do not exactly match the reference text.

A.4 Implementation Details

All experiments were implemented using the HuggingFace Transformers library with a PyTorch backend, which provided a flexible and reliable framework for working with pre-trained QA models. Inputs were tokenized using the respective model tokenizers, and the maximum input length was set to accommodate the chosen context sizes, ensuring that the models could process all passages in each instance without truncation.

The models were run in inference mode on a GPU-enabled environment on Colab, which allowed us to efficiently handle the large number of passages and maintain reasonable processing times. Using Colab provided a convenient and reproducible platform, with consistent hardware and software configurations.

A.5 Evaluation Results

Tables 1–5 report the detailed Exact Match (EM) and F1 scores of GPT-4.1-nano and Gemini-2.5-Flash-Lite across different context lengths (5, 10, 20, and 30) and varying positions of the relevant passage within the input.

The results show a consistent positional bias: performance is highest when the relevant context is placed at the beginning of the input, declines significantly when the context is in the middle, and recovers slightly when it appears at the end. This U-shaped trend aligns with the "lost in the middle" phenomenon observed in prior long-context QA studies, suggesting that the effect also holds for Bangla question answering.

Table 3: Evaluation results of GPT-4.1-nano and Gemini-2.5-Flash-Lite with context length 5 at different indices, showing Exact Match (EM) and F1 Score.

Index	GPT-4.1-nano		Gemini-2.5-flash-lite	
	EM	F1	EM	F1
0	64.13%	69.42%	60.63%	75.66%
3	44.76%	51.52%	59.68%	74.50%
5	49.52%	55.04%	60.95%	75.45%

Table 4: Evaluation results of GPT-4.1-nano and Gemini-2.5-Flash-Lite with context length 20 at different indices, showing Exact Match (EM) and F1 Score.

Index	GPT-4.1-nano		Gemini-2.5-flash-lite	
	EM	F1	EM	F1
0	59.37%	67.07%	59.37%	74.17%
5	45.08%	52.13%	54.60%	69.27%
10	36.51%	43.07%	53.97%	68.49%
15	34.92%	40.27%	47.94%	64.49%
20	37.78%	42.91%	55.56%	70.93%

Figure 3- 5 shows the performance of both models on different context lengths and gold passage positions.

Table 5: Evaluation results of GPT-4.1-nano and Gemini-2.5-Flash-Lite with context length 30 at different indices, showing Exact Match (EM) and F1 Score.

Index	GPT-4.1-nano		Gemini-2.5-flash-lite	
	EM	F1	EM	F1
1	58.73%	64.96%	59.37%	74.41%
5	42.22%	47.78%	52.70%	67.31%
10	33.97%	39.62%	49.52%	65.74%
15	35.56%	41.56%	44.60%	59.45%
20	32.06%	36.05%	45.40%	60.09%
25	30.16%	34.09%	46.67%	60.84%
30	35.24%	39.69%	52.06%	67.13%

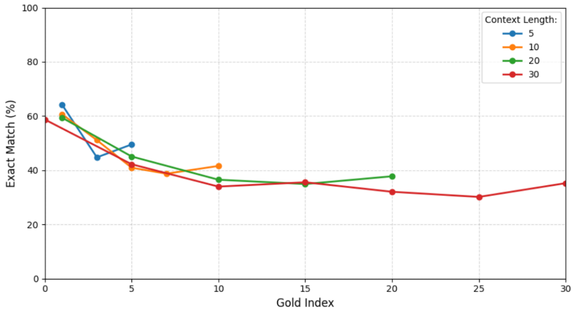


Figure 3: Performance comparison using Exact Match in GPT-4.1-nano

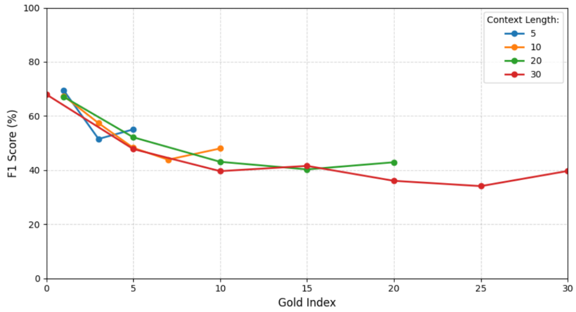


Figure 4: Performance comparison using F1 Score in GPT-4.1-nano

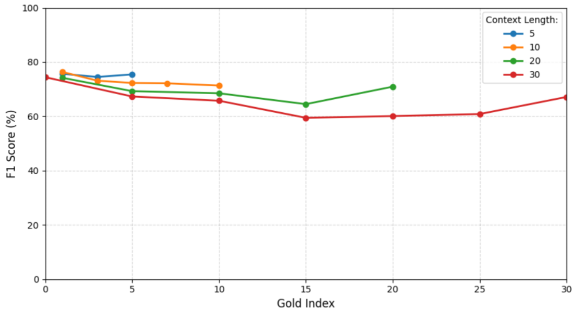


Figure 5: Performance comparison using F1 Score in Gemini-2.5-Flash-Lite

Form-aware Poetic Generation for Bangla

Amina, Abdullah, Mueeze Al Mushabbir, Sabbir Ahmed
Department of Computer Science and Engineering
Islamic University of Technology, Bangladesh
{amina, abdullah20, almushabbir, sabbirahmed}@iut-dhaka.edu

Abstract

Poetry generation in low-resource languages such as Bangla is particularly challenging due to the scarcity of structured poetic corpora and the complexity of its metrical system (*matra*). We present a structure-aware framework for Bangla poetry generation using pretrained Bangla large language models (LLMs)–TigerLLM, TituLLM, and BanglaT5–trained on general non-poetic text corpora augmented with rich structural control tokens. These tokens capture rhyme, meter, word count, and line boundaries, enabling unsupervised modeling of poetic form without curated poetry datasets. Unlike prior fixed-pattern approaches, our framework introduces variable control compositions, allowing models to generate flexible poetic structures. Experiments show that explicit structural conditioning improves rhyme consistency and metrical balance while maintaining semantic coherence. Our study provides the first systematic evaluation of Bangla LLMs for form-constrained creative generation, offering insights into structural representation in low-resource poetic modeling.

1 Introduction

Poetry generation balances semantic fluency with formal constraints such as rhyme, meter, and line length (Mahbub et al., 2023; Hu et al., 2024). For low-resource languages like Bangla, this is challenging due to scarce curated poetic corpora and a complex *matra*-based prosody (Pakray et al., 2025).

Prior work, such as PoeLM (Ormazabal et al., 2022), showed that transformer LMs can learn rhyme and metrical patterns from general corpora augmented with structural codes, but relied on fixed configurations and focused on Spanish and Basque.

Bangla's phonetic and morphological complexity, coupled with limited poetic datasets and pretrained LLMs evaluated mainly on prose, motivates our use of non-poetic text with variable control tokens (<RYM>, <MTR>, <WRD>, <STA>, <FIN>) to enable structure-aware poetry generation.

Our contributions are: (1) a framework for unsupervised Bangla poetry generation with variable control tokens; (2) systematic evaluation of pretrained Bangla LLMs (TigerLLM (Raihan and Zampieri, 2025), TituLLM (Nahin et al., 2025), BanglaT5 (Bhattacharjee et al., 2023)) on form-constrained generation; (3) insights into how structural conditioning improves adherence to poetic form while maintaining coherence.

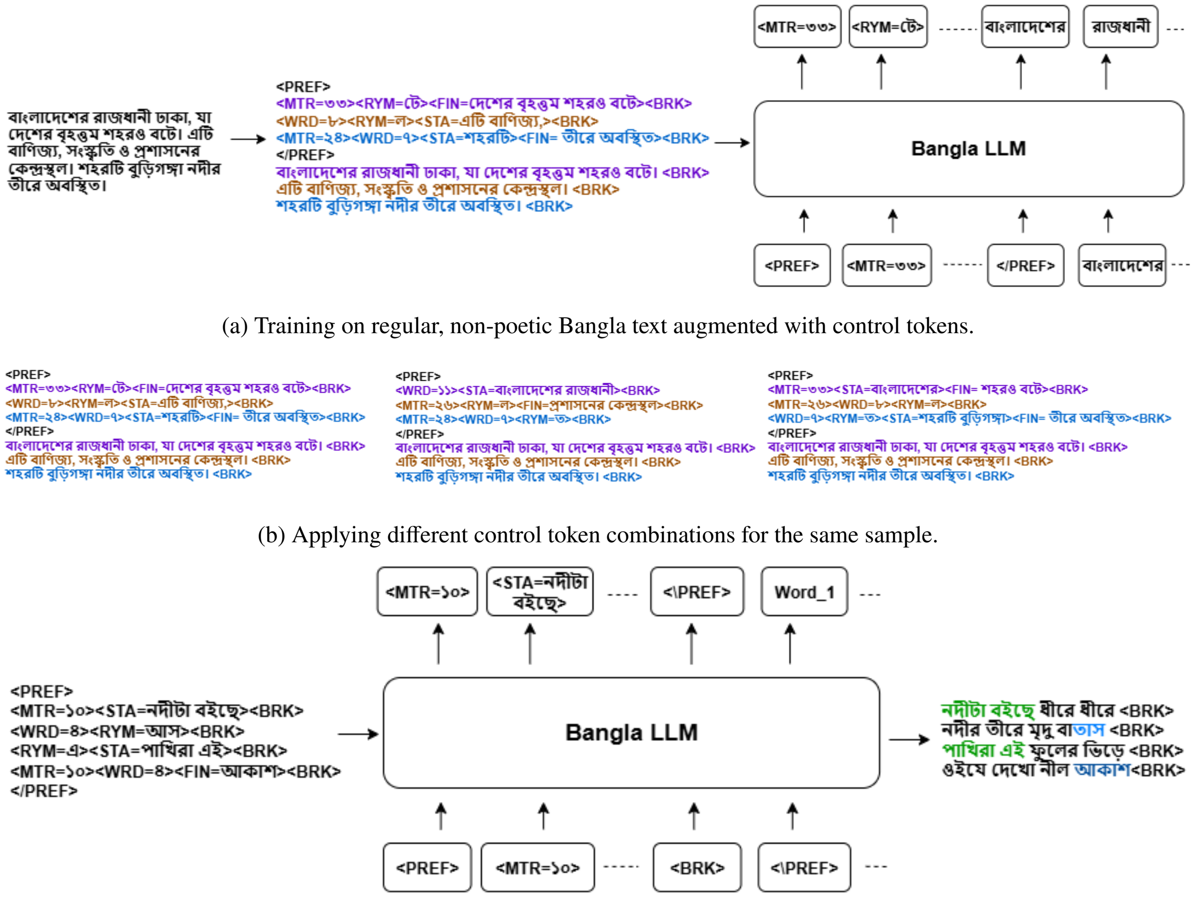
We show that pretrained Bangla LLMs can generate metrically consistent, rhymed, and semantically coherent verses under structural guidance, providing a foundation for further research in this domain.

2 Related Works

Modern advances in large language models (LLMs) have significantly reshaped natural language processing, enabling systems to achieve strong performance across tasks ranging from text generation and summarization to reasoning and dialogue (Abrar et al., 2024; Arif et al., 2025; Khan et al., 2023b; Ahmed et al., 2024; Khan et al., 2023a). These developments have also opened new possibilities for creative text generation, where models learn not only linguistic fluency but also stylistic, structural, and domain-specific patterns (Shedko, 2018; Gonçalo Oliveira, 2024).

Poetry generation in NLP has evolved from rule-based and template systems (Das, 2014), which enforced form but lacked fluency, to neural approaches using LSTMs and transformers for both free and structured verse (Ghazvininejad et al.,

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 366–372
December 23, 2025 ©2025 Association for Computational Linguistics


বাংলাদেশের রাজধানী ঢাকা, যা দেশের বৃহত্তম শহরও বটে। এটি বাণিজ্য, সংস্কৃতি ও প্রশাসনের কেন্দ্রস্থল। শহরটি বুড়িগঙ্গা নদীর তীরে অবস্থিত।
<PREF>
<MTR=৩৩><RYM=টে><FIN=দেশের বৃহত্তম শহরও বটে><BRK>
<WRD=৮><RYM=ল><STA=এটি বাণিজ্য,><BRK>
<MTR=২৪><WRD=৭><STA=শহরটি><FIN= তীরে অবস্থিত><BRK>
</PREF>
বাংলাদেশের রাজধানী ঢাকা, যা দেশের বৃহত্তম শহরও বটে। <BRK>
এটি বাণিজ্য, সংস্কৃতি ও প্রশাসনের কেন্দ্রস্থল। <BRK>
শহরটি বুড়িগঙ্গা নদীর তীরে অবস্থিত। <BRK>
Bangla LLM
<MTR=৩৩>
<RYM=টে>
বাংলাদেশের
রাজধানী
<PREF>
<MTR=৩৩>
</PREF>
বাংলাদেশের
<PREF>
<MTR=৩৩><RYM=টে><FIN=দেশের বৃহত্তম শহরও বটে><BRK>
<WRD=৮><RYM=ল><STA=এটি বাণিজ্য,><BRK>
<MTR=২৪><WRD=৭><STA=শহরটি><FIN= তীরে অবস্থিত><BRK>
</PREF>
বাংলাদেশের রাজধানী ঢাকা, যা দেশের বৃহত্তম শহরও বটে। <BRK>
এটি বাণিজ্য, সংস্কৃতি ও প্রশাসনের কেন্দ্রস্থল। <BRK>
শহরটি বুড়িগঙ্গা নদীর তীরে অবস্থিত। <BRK>
<PREF>
<WRD=১১><STA=বাংলাদেশের রাজধানী><BRK>
<MTR=২৬><RYM=ল><FIN=প্রশাসনের কেন্দ্রস্থল><BRK>
<MTR=২৪><WRD=৭><RYM=ত><BRK>
</PREF>
বাংলাদেশের রাজধানী ঢাকা, যা দেশের বৃহত্তম শহরও বটে। <BRK>
এটি বাণিজ্য, সংস্কৃতি ও প্রশাসনের কেন্দ্রস্থল। <BRK>
শহরটি বুড়িগঙ্গা নদীর তীরে অবস্থিত। <BRK>
<PREF>
<MTR=৩৩><STA=বাংলাদেশের><FIN= শহরও বটে><BRK>
<MTR=২৬><WRD=৮><RYM=ল><BRK>
<WRD=৭><RYM=ত><STA=শহরটি বুড়িগঙ্গা><FIN= তীরে অবস্থিত><BRK>
</PREF>
বাংলাদেশের রাজধানী ঢাকা, যা দেশের বৃহত্তম শহরও বটে। <BRK>
এটি বাণিজ্য, সংস্কৃতি ও প্রশাসনের কেন্দ্রস্থল। <BRK>
শহরটি বুড়িগঙ্গা নদীর তীরে অবস্থিত। <BRK>
<MTR=১০>
<STA=নদীটা বইছে>
<\PREF>
Word_1
<PREF>
<MTR=১০><STA=নদীটা বইছে><BRK>
<WRD=8><RYM=আস><BRK>
<RYM=এ><STA=পাখিরা এই><BRK>
<MTR=১০><WRD=8><FIN=আকাশ><BRK>
</PREF>
Bangla LLM
নদীটা বইছে ধীরে ধীরে <BRK>
নদীর তীরে মৃদু বাতাস <BRK>
পাখিরা এই ফুলের ভিড়ে <BRK>
ওইযে দেখো নীল আকাশ<BRK>
<PREF>
<MTR=১০>
<BRK>
<\PREF>

(a) Training on regular, non-poetic Bangla text augmented with control tokens.

(b) Applying different control token combinations for the same sample.

(c) Controlled generation under mixed guidance tokens.

Figure 1: Overview of the proposed Bangla poetry generation framework: (a) During training, regular Bangla text is segmented and augmented with structural control tokens, enabling the model to associate textual patterns with explicit poetic constraints. (b) At inference time, multiple combinations of control tokens are applied to the same input, allowing flexible specification of desired poetic form. (c) The fine-tuned model generates verse conditioned on these tokens, producing lines that follow the requested structural layout while maintaining semantic continuity across the poem.

2016; Yang et al., 2023). These methods typically require large annotated poem corpora, limiting applicability in low-resource languages (Zhong et al., 2025).

In Bangla, Murad and Rahman (2023) and Chy et al. (2020) leveraged deep learning-based techniques to generate poet-style free-form poems but did not systematically enforce meter or rhyme. Ormazabal et al. (2022) introduced 'PoeLM', an unsupervised controlled verse generation with structural tokens for Spanish and Basque, yet relied on fixed patterns and simpler poetic attributes. Recent LLMs tuned for Bangla language, including TigerLLM (Raihan and Zampieri, 2025), TituLLMs (Nahin et al., 2025), and BanglaT5 (Bhattacharjee et al., 2023), have mostly been evaluated on prose. Studies on Bangla meter (Ahmed et al., 2023) provide a basis for controlled generation.

Our work extends these ideas by using variable control tokens for rhyme, meter, word count, and line boundaries, enabling flexible, unsupervised Bangla poetry generation from non-poetic corpora and evaluation of pretrained LLMs in this setting.

3 Methodology

3.1 Overview

Our work extends PoeLM (Ormazabal et al., 2022) to Bangla by developing a structure-aware poetry generation framework, shown in Figure 1, that uses control tokens for rhyme, meter, word count, and line boundaries. Instead of curated poetry datasets, we repurpose general Bangla text into poem-like training samples with these tokens, enabling unsupervised learning of poetic form and generation of verses that preserve rhythmic consistency and semantic coherence under flexible structural guidance.

3.2 Corpus Acquisition and Preprocessing

Due to the scarcity of annotated Bangla poetry corpora, we used the Bangla portion of the OSCAR 2019 corpus (OSCAR bn) (Ortiz Suárez et al., 2020). OSCAR 2019 was generated from the Common Crawl using the goclassy architecture, containing noisy text from multiple languages. The Bangla sub-corpus contains both original and deduplicated versions. We use the deduplicated portion, which has 363,766,143 words (5.8 GB), to ensure that repeated lines do not bias model training.

The raw text is further cleaned to remove non-Bangla content, extraneous symbols, and malformed sentences. From the cleaned corpus, we extract coherent multi-line segments to serve as poem-like units. Sample creation follows these steps:

1. Choose a random starting point in corpus.
2. Select a variable number of consecutive lines for the segment.
3. Vary word lengths within each line randomly, avoiding single-word lines.
4. Assign structural control tokens to each multi-line segment.

This process preserves local context and allows the model to learn metrically and rhymically coherent text across multiple lines.

3.3 Control Tokens

Each multiline segment is annotated with control tokens to provide explicit structural guidance:

- **Rhyme tokens** (<RYM=...>): encode the rhyme class of the last syllable.
  Example: <RYM=অয়> → line must end with rhyme "অয়".
- **Meter tokens** (<MTR=...>): specify the number of meter/*matra* in a line.
  Example: <MTR=২৬> → line must contain 26 *matra*.
- **Word count tokens** (<WRD=...>): enforce the target number of words.
  Example: <WRD=৯> → line must contain exactly 9 words.
- **Start** (<STA=...>) **and End** (<FIN=...>) **tokens**: encode line fragments for guidance.
  Example: <STA= আমি আজ> → line must start with "আমি আজ ...",
  <FIN= ভোরের আলোয়> → line must end with "... ভোরের আলোয়".

3.4 Adaptive Control Token Schema

Control tokens are embedded directly into the training text, allowing the model to learn them as part of its vocabulary (Figure 1a). Unlike fixed-pattern approaches such as PoeLM (Ormazabal et al., 2022), each training instance can include any subset of available tokens–rhyme, meter, word count, start, and end; covering all possible combinations.

Exposure to diverse token combinations encourages generalization across varying levels of structural guidance (Figure 1b). This design supports generation under full, partial, or mixed prompts while preserving semantic coherence and poetic form.

3.5 Model Training

We adopt a next-token prediction approach where the model is conceptually conditioned on control tokens and optional start/end fragments. This allows the model to learn form-aware generation and handle flexible prompts, supporting rhyme, meter, and word count constraints at a high level.

3.6 Generation with Filtered Re-ranking

The generation process happens in three steps to ensure structural adherence and fluency:

Candidate Generation: The trained model generates multiple candidate verses conditioned on user-specified control tokens. Sampling strategies like top-k or nucleus sampling are used to ensure diversity.

Structural Filtering: Candidates are filtered to retain only those satisfying the structural constraints:

- Word count matches the target.
- Meter (*matra*) approximates the specified value.
- Rhyme ends with the required syllable.
- Start and end fragments are respected.
- Repetition of rhyming words is avoided.

Re-ranking: Filtered candidates are scored for fluency using model likelihoods. The highest-scoring candidate, balancing structural correctness and semantic coherence, is selected as the final output.

4 Experimental Setup

4.1 Training Strategy

We fine-tuned three pretrained Bangla LLMs–TigerLLM (Raihan and Zampieri, 2025), TituLLM (Nahin et al., 2025), and BanglaT5 (Bhattacharjee et al., 2023)–on descriptor-augmented multi-line segments. Each training sample consisted of a prefix of control tokens and optional start/end fragments, with the model predicting the remaining tokens.

4.2 Evaluation Metrics

We measured both structural adherence and semantic coherence:

- **Rhyme correctness**: whether generated lines end with the specified rhyme.
- **Word-count correctness**: whether the number of words matches the target.
- **Meter correctness**: whether syllable counts match the specified *matra*.
- **Start/End correctness**: whether start and end fragments are respected.
- **Overall structure following**: whether all structural constraints are consistently met across the poem.
- **Text coherence**: grammatical validity and semantic consistency.

Rhyme, meter, word-count, start/end correctness, and overall structure following were evaluated using automatic validators, while coherence was assessed through manual qualitative inspection.

4.3 Human Evaluation Criteria

To complement the large-scale automatic structure estimates, we conducted a focused human evaluation to assess the coherence of generated poems. Three native Bangla speakers with experience reading both contemporary and classical Bangla poetry served as annotators. A stratified sample of model outputs was constructed to cover all model variants. Annotators independently rated each poem along the two components of coherence defined in our evaluation protocol:

- **Grammatical validity**: syntactic well-formedness and naturalness of phrasing.
- **Semantic consistency**: clarity of meaning within and across lines, and whether poem maintains a coherent thematic flow.

Both criteria were rated on a 5-point Likert scale. Written guidelines ensured a shared interpretation of the criteria. Samples were divided among annotators with partial overlap to allow reliability estimation; ratings were collected independently and averaged. Inter-annotator consistency was checked using a standard reliability measure for ordinal judgments and showed reasonable agreement. For reporting, we compute a **combined human coherence score** by averaging the grammatical and semantic ratings.

5 Results and Analysis

5.1 Quantitative Evaluation

Table 1 reports both baseline and finetuned performance of pretrained Bangla LLMs across structural and semantic metrics. Scores denote the percentage of generated lines satisfying each constraint (rhyme, word-count, meter, and boundary tokens), along with overall textual coherence.

Across all models, baseline scores remain low for structural metrics, reflecting that pretrained LLMs–without control-token finetuning–cannot reliably follow poetic form. However, coherence remains relatively high even in baseline generations, indicating that the models already possess strong linguistic fluency. Finetuning with structural control tokens yields substantial improvements: TigerLLM shows the strongest gains, achieving over 90% accuracy on all structural categories. TituLLM also improves consistently, though with slightly lower meter precision. BanglaT5 benefits from finetuning as well, but lags behind the larger models in structure fidelity. Overall, these results demonstrate that explicit structural conditioning is highly effective in enabling Bangla LLMs to generate form-consistent poetry.

5.2 Qualitative Evaluation

Qualitative inspection shows that variable control-token fine-tuning helps maintain coherence across lines, ensuring each verse flows naturally while respecting the specified rhyme, meter, word-count, and boundary constraints. Baseline generations, by contrast, often produce disjointed lines that, although locally fluent, fail to maintain continuity across the segment. Table 2 presents high-ranked

Table 1: Comparison of baseline (no control-token fine-tuning) and control-token–fine-tuned versions of each pretrained Bangla LLM.

Model / Setting	**Rhyme**	**Word-count**	**Meter**	**Start/End**	**Structure**	**Coherence**
TigerLLM (baseline)	42%	61%	45%	50%	51%	86%
TigerLLM (finetuned)	95%	97%	93%	96%	95%	94%
TituLLM (baseline)	28%	48%	33%	36%	36%	77%
TituLLM (finetuned)	93%	95%	88%	94%	91%	90%
BanglaT5 (baseline)	25%	42%	28%	32%	31%	72%
BanglaT5 (finetuned)	85%	87%	80%	82%	83%	88%

verse generated by each model after filtered reranking from prompts illustrated in Figure 2.

আমাকে তিন লাইনের একটি কবিতা লিখে দাও যার, প্রথম লাইনের মোট মাত্রা সংখ্যা ১০, লাইনের শেষ অংশ 'ঘাস'; দ্বিতীয় লাইনের মোট মাত্রা সংখ্যা ১০, লাইনের শুরুর অংশ 'চোখ', লাইনের শেষ অংশ 'চারপাশ'; তৃতীয় লাইনের শেষ ছন্দ 'আস', মোট শব্দ সংখ্যা ৫।

(a) Prompt format for baseline models

```
<PREF>
<MTR=১০><FIN=ঘাস><WRD=৫><BRK>
<MTR=১০><STA=চোখ><FIN=চারপাশ><BRK>
<RYM=আস><WRD=৫><BRK>
</PREF>
```

(b) Prompt format for fine-tuned models

Figure 2: Example of a test sample for qualitative evaluation.

Table 2: High-ranked verse generated from each baseline (-bs) and fine-tuned (-ft) Bangla LLMs for sample prompts in Figure 2.

Model	**Sample Generated Verse**
TigerLLM-bs	আসে রোশনির চারপাশ করে সবুজ ঘাস, চোখ দিয়ে দেখি সূর্যের চারিপাশ, চারপাশের ঘাসে লুকিয়ে আছে আনন্দ।
TigerLLM-ft	সকালের রোদে ভিজে ওঠে ঘাস, চোখ রেখে দেখি নরম চারপাশ আজ বাতাসে ভাসে নতুন উচ্ছ্বাস।
TituLLM-bs	তিন লাইনের একটি কবিতা শেষে ঘাস, চোখ দিয়ে শুরু মাত্রা দশ শেষে ছন্দ আস আস আস।
TituLLM-ft	পাহাড়ের ঢালে সবুজ নরম ঘাস চোখ মেলে দেখি চারপাশ, হাওয়ায় ভাসে শান্ত নীল বিশ্বাস।
BanglaT5-bs	মাত্রা মাত্রা ঘাস লিখো চোখ লিখো চারপাশ লিখো আস ছন্দ শব্দ লিখো
BanglaT5-ft	সকালের শিশির ভেজা নরম ঘাসে। চোখে দেখি দূরের পাহাড়। আজ বাতাস খুব নরম ঘাস।

These examples highlight that control tokens not only enforce formal constraints but also support semantic connectedness across lines, demonstrating their utility for producing coherent Bangla poetry beyond mere structural compliance.

6 Conclusion

We presented a framework to generate Bangla poetry guided by control tokens for rhyme, meter, word count, and line boundaries. Our models achieve high structural adherence while maintaining semantic coherence by fine-tuning pretrained Bangla LLMs on descriptor-augmented multi-line segments. Both quantitative and qualitative evaluations confirm the effectiveness of our approach, highlighting the potential of control-token-guided generation for creative text. Future work could focus on improving context-awareness, enabling finer-grained control, and better balancing structural fidelity with semantic coherence.

7 Limitations

While our framework successfully generates form-aware Bangla poetry that adheres to rhyme, meter, and word-count constraints, several limitations remain. Controllability is still imperfect: the model may drift from specified constraints when multiple or conflicting control tokens are applied, and longer generations can exhibit weakened semantic coherence or fragmented narrative flow. Moreover, although the system effectively enforces structural attributes, it does not explicitly capture deeper stylistic properties—such as tone, imagery, or poet-specific voice—which can lead to verses that feel less expressive or stylistically homogeneous. Finally, since our work builds incrementally on the ideas introduced in PoeLM (Ormazabal et al., 2022), a more comprehensive investigation comparing stylistic behavior and constraint robustness across both approaches remains an important direction for future work.

References

Ajwad Abrar, Farzana Tabassum, and Sabbir Ahmed. 2024. Performance evaluation of large language models in bangla consumer health query summarization. In *2024 27th International Conference on Computer and Information Technology (ICCIT)*, pages 2748–2753.

Nasim Ahmed, Sheikh Taraque Aziz, Md. Abu Naser Mojumder, and Maruf Ahmed Mridul. 2023. Automatic classification of meter in bangla poems: A machine learning approach. In *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, pages 1–5.

Tasnim Ahmed, Shahriar Ivan, Ahnaf Munir, and Sabbir Ahmed. 2024. Decoding depression: Analyzing social network insights for depression severity assessment with transformers and explainable ai. *Natural Language Processing Journal*, 7:100079.

Nokimul Hasan Arif, Shadman Rabby, Md Hefzul Hossain Papon, and Sabbir Ahmed. 2025. Preemptive hallucination reduction: An input-level approach for multimodal language model. *Preprint*, arXiv:2505.24007.

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.

Md. Kalim Amzad Chy, Md. Abdur Rahman, Abdul Kadar Muhammad Masum, Shayhan Ameen Chowdhury, Md. Golam Robiul Alam, and Md. Shahidul Islam Khan. 2020. Bengali poem generation using deep learning approach. In *Intelligent Computing Paradigm and Cutting-edge Technologies*, pages 148–157.

Amitava Das. 2014. Poetic machine: Computational creativity for automatic poetry generation. In *Proceedings of the 2014 International Conference on Computational Creativity*, pages 11–18.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin, Texas. Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2024. Automatic generation of creative text in portuguese: an overview. *Language Resources and Evaluation*, 58(1):7–41.

Zhiyuan Hu, Chumin Liu, Yue Feng, Anh Tuan Luu, and Bryan Hooi. 2024. Poetrydiffusion: Towards joint semantic and metrical manipulation in poetry generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18279–18288.

Alvi Khan, Fida Kamal, Mohammad Abrar Chowdhury, Tasnim Ahmed, Md Tahmid Rahman Laskar, and Sabbir Ahmed. 2023a. BanglaCHQ-summ: An abstractive summarization dataset for medical queries in Bangla conversational speech. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 85–93, Singapore. Association for Computational Linguistics.

Alvi Khan, Fida Kamal, Nuzhat Nower, Tasnim Ahmed, Sabbir Ahmed, and Tareque Chowdhury. 2023b. NERvous about my health: Constructing a Bengali medical named entity recognition dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5768–5774, Singapore. Association for Computational Linguistics.

Ridwan Mahbub, Ifrad Khan, Samiha Anuva, Md Shihab Shahriar, Md Tahmid Rahman Laskar, and Sabbir Ahmed. 2023. Unveiling the essence of poetry: Introducing a comprehensive dataset and benchmark for poem summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14878–14886, Singapore. Association for Computational Linguistics.

Hasan Murad and Rashik Rahman. 2023. Ai poet: A deep learning based approach to generate artificial poetry in bangla. In *Applied Intelligence for Industry 4.0*. CRC Press.

Shahriar Kabir Nahin, Rabindra Nath Nandi, Sagor Sarker, Quazi Sarwar Muhtaseem, Md Kowsher, Apu Chandraw Shill, Md Ibrahim, Mehadi Hasan Menon, Tareq Al Muntasir, and Firoj Alam. 2025. TituLLMs: A family of Bangla LLMs with comprehensive benchmarking. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24922–24940, Vienna, Austria. Association for Computational Linguistics.

Aitor Ormazabal, Mikel Artetxe, Manex Agirrezabal, Aitor Soroa, and Eneko Agirre. 2022. Poelm: A meter- and rhyme-controllable language model for unsupervised poetry generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 268–278.

Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2020. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online. Association for Computational Linguistics.

Partha Pakray, Alexander Gelbukh, and Sivaji Bandyopadhyay. 2025. Natural language processing applications for low-resource languages. *Natural Language Processing*, 31(2):183–197.

Nishat Raihan and Marcos Zampieri. 2025. TigerLLM - a family of Bangla large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 887–896, Vienna, Austria. Association for Computational Linguistics.

Andrey Y. Shedko. 2018. Semantic-map-based assistant for creative text generation. *Procedia Computer Science*, 123:446–450. 8th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2017 (Eighth Annual Meeting of the BICA Society), held August 1-6, 2017 in Moscow, Russia.

Liang Yang, Zhexu Shen, Fengqing Zhou, Hongfei Lin, and Junpeng Li. 2023. Tpoet: Topic-enhanced chinese poetry generation. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 22(6).

Tianyang Zhong, Zhenyuan Yang, Zhengliang Liu, Ruidong Zhang, Yiheng Liu, Haiyang Sun, Yi Pan, Yiwei Li, Yifan Zhou, Hanqi Jiang, Junhao Chen, and Tianming Liu. 2025. Opportunities and challenges of large language models for low-resource languages in humanities research. *Preprint*, arXiv:2412.04497.

Overview of BLP-2025 Task 2: Code Generation in Bangla

Nishat Raihan[1], **Mohammad Anas Jawad**[2], **Md Mezbaur Rahman**[2],
Noshin Ulfat[3], **Pranav Gupta**[5], **Mehrab Mustafy Rahman**[2],
Shubhra Kanti Karmakar[4], **Marcos Zampieri**[1]

[1]George Mason University, [2]University of Illinois Chicago, [3]IQVIA,
[4]University of Central Florida, [5]Lowe's

mraihan2@gmu.edu

Abstract

This paper presents an overview of the BLP 2025 shared task **Code Generation in Bangla**[1], organized with the BLP workshop co-located with AACL. The task evaluates Generative AI systems capable of generating executable Python code from natural language prompts written in Bangla. This is the first shared task to address Bangla code generation. It attracted 152 participants across 63 teams, yielding 488 submissions, with 15 system-description papers. Participating teams employed both proprietary and open-source LLMs, with prevalent strategies including prompt engineering, fine-tuning, and machine translation. The top `Pass@1` reached 0.99 on the development phase and 0.95 on the test phase. In this report, we detail the task design, data, and evaluation protocol, and synthesize methodological trends observed across submissions. Notably, we observe that the high performance is not based on single models; rather, a pipeline of multiple AI tools and/or methods.

1 Introduction

Despite being the world's fifth most spoken language, Bangla remains underrepresented in Large Language Models (LLMs)—especially for code generation, even as recent advances markedly improve code synthesis (Touvron et al., 2023; Hui et al., 2024a; Team et al., 2025). State-of-the-art (SOTA) models now exceed 90% `Pass@1` on prominent benchmarks such as HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), spurring adoption in software engineering (Pasquale et al., 2025) and education (Raihan et al., 2025c). Yet these gains disproportionately accrue to a few high-resource languages (Joshi et al., 2020; Blasi et al., 2022; Ahuja et al., 2023; Wang et al., 2023; Raihan et al., 2024).

Bangla—spoken by over 242 million native speakers[2], still lacks dedicated code-generation resources: datasets are scarce, tooling is limited, and benchmarks are largely absent (Bhattacharjee et al., 2022; Zehady et al., 2024). Consequently, general-purpose Bangla models are outperformed by their English counterparts on code-related tasks (Bhattacharyya et al., 2023; Uddin et al., 2023), underscoring the need for targeted data, evaluation suites, and modeling efforts.

While Bangla Natural Language Understanding (NLU) and Generation (NLG) see considerable growth with resources like BanglaRQA (Ekram et al., 2022) and BEnQA (Shafayat et al., 2024), the domain of code generation remains relatively under-explored. Prior work in this area is limited to two main benchmarks: `mHumanEval-Bangla` (Raihan et al., 2025a), a subset of a multilingual evaluation benchmark containing 164 prompts adapted from the HumanEval dataset, and `MBPP-Bangla` (Raihan et al., 2025b), which provides 974 coding prompts adapted from the MBPP dataset. For this shared task, we utilize a combined dataset composed of both `mHumanEval-Bangla` and `MBPP-Bangla`.

Our motivation for this shared task is to improve the performance of Bangla NLP models on code generation. The primary objective is to introduce a more advanced task that evaluates the emerging code generation capabilities of LLMs. As the first task of its kind for

[1] Task website: https://noshinulfat.github.io/blp25_code_generation_task/#/home

[2] https://www.ethnologue.com/

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 373–387
December 23, 2025 ©2025 Association for Computational Linguistics

Bangla, we provide extensive support to participants, including a starter kit[3], tutorials, and seminars. Participants are also granted the flexibility to use any proprietary or open-source models, alongside any NLP methods. This open approach is intended not only to provide a strong starting point but also to uncover diverse strategies for solving a new and complex task for LLMs in a mid-resource language, yielding key insights for future research.

We elaborate on the task and present our findings, the remainder of this paper is organized as follows: Section 2 discusses the datasets used during the task, Section 3 describes the task and the it's two phases (dev & test), Section 4 includes the participants' results, Section 6 summarizes the approaches taken by the system description papers and Section 6 investigates the key insights.

2 Data

We utilize the only two available benchmarks for Bangla Code Generation: `mHumanEval-Bangla` (Raihan et al., 2025a) and `MBPP-Bangla` (Raihan et al., 2025b). These are selected for their distinct and complementary qualities.

Specs	HumanEval-Bangla	MBPP-Bangla
# of Tasks	164	974
Prompt	Bangla	Bangla
Solution	Python	Python
Problem source	Hand-written	Crowd-sourced
Task focus	Function completion	Basic–intermediate
Problem format	Docstring	Short prompt
Tests per task	7.7 (avg.)	3
Metric	pass@1	pass@1

Table 1: Dataset details for `HumanEval-Bangla` and `MBPP-Bangla`.

`MBPP-Bangla` offers scale and breadth: its 974 short, crowd-sourced Bangla prompts yield more coverage, which estimates and stress a model's ability to handle a wide variety of basic–intermediate tasks. `HumanEval-Bangla` complements this with depth: 164 hand-written, docstring-based function-completion problems paired with denser test suites (7.7 test cases on avg.) vs. 3 tests per task) probe precise adherence to specification. Evaluating on both benchmarks provides a more detailed picture of Bangla-to-Python code generation—breadth and robustness from `MBPP-Bangla`, and precision and rigor from `HumanEval-Bangla`. We have made the combined version publicly available. [4]

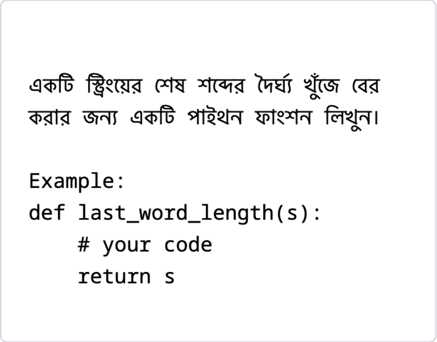
3 Task Description

In this task, we evaluate LLMs on one of their emerging capabilities, code generation. The task becomes more challenging as the prompts used in our task are in Bangla. As mentioned before, this is the first shared task of its kind in the Bangla NLP domain.

In formal definition, the task entails:

> Given a set of coding prompts (task descriptions and/or docstrings) in Bangla, the participants will have to use (prompt, finetune, etc.) LLMs to generate corresponding Python code snippets that pass all the test cases for that particular task. The evaluation metric is `Pass@1`, meaning that the models will have only one attempt to pass all the test cases for a particular prompt.

Task examples from both benchmarks are shown in Figure 1 and 2. We launch the dev phase of the task on the Codabench [5] platform on August 10th, 2025.



```
একটি স্ট্রিংয়ের শেষ শব্দের দৈর্ঘ্য খুঁজে বের
করার জন্য একটি পাইথন ফাংশন লিখুন।

Example:
def last_word_length(s):
    # your code
    return s
```

Figure 1: Sample prompt from MBPP-Bangla. Translation: *'Write a Python function to find the length of the last word in a given string'.*

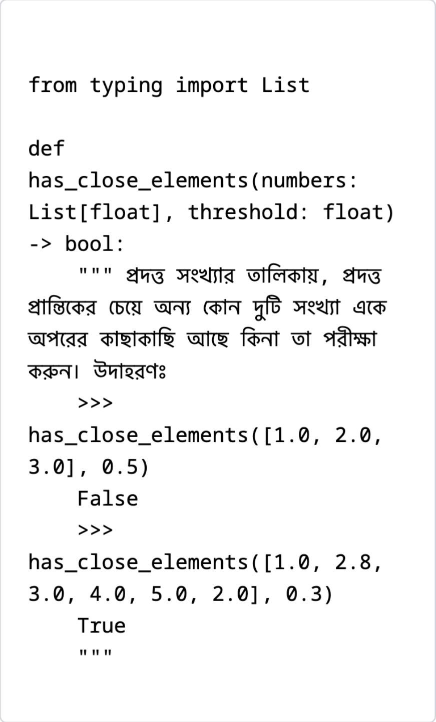
[3] Starter Kit: https://noshinulfat.github.io/blp25_code_generation_task/#/starter-kit

[4] Datasets: https://noshinulfat.github.io/blp25_code_generation_task/#/starter-kit

[5] Competition website: https://www.codabench.org/competitions/10089/

Specification	DEV	TEST	Total
Start Date	August 10, 2025	September 7, 2025	—
End Date	September 8, 2025	September 14, 2025	—
Duration	28 Days	7 Days	35 Days
Participants	152	97	152
Teams	63	32	63
Submissions	301	187	488
Average (Submission)	4.78	5.84	7.75
Test Cases	Fully Released	One per Task	—
Highest Score (Pass@1)	0.99	0.95	—
Lowest Score	0.00	0.02	—
Average Score	0.52	0.59	—

Table 2: DEV/TEST timeline and participation summary. *Average (Submission)* denotes average submissions per team.



```
from typing import List

def
has_close_elements(numbers:
List[float], threshold: float)
-> bool:
    """ প্রদত্ত সংখ্যার তালিকায়, প্রদত্ত
প্রান্তিকের চেয়ে অন্য কোন দুটি সংখ্যা একে
অপরের কাছাকাছি আছে কিনা তা পরীক্ষা
করুন। উদাহরণঃ
    >>>
has_close_elements([1.0, 2.0,
3.0], 0.5)
    False
    >>>
has_close_elements([1.0, 2.8,
3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
```

Figure 2: Sample prompt from mHumanEval-Bangla. Translation: *'Write a Python function to find that if two numbers in a list are closer to each other than the given threshold'*.

3.1 DEV Phase

In the DEV phase, systems operate under high observability: fully released test cases and a longer horizon enable targeted debugging and steady pipeline stabilization. Teams submit at a disciplined rate (Avg. 4.78 submissions per team), and performance spans a wide range, from 0.00 to a near-ceiling 0.99 Pass@1 (Table 2). The broad spread, together with an average score of 0.52, indicates heterogeneous readiness—strong systems rapidly approach the ceiling, while weaker pipelines expose specification and edge-case errors that visible tests help uncover.

Methodologically, DEV functions as an internal-validity probe: with rich feedback, improvements reflect engineering rigor and prompt–test alignment rather than guesswork. The combination of high best score and moderate average suggests a bimodal landscape in which top teams consolidate gains early while others iterate to resolve stability issues. These dynamics make DEV well-suited for ablations and reproducibility checks, as changes map cleanly onto observable error reductions (Table 2).

3.2 TEST Phase

The TEST phase tightens observability—one visible test per task over a shorter window—shifting the emphasis from iterative debugging to generalization under uncertainty. Teams react by concentrating effort: average submissions per team increases to 5.84 despite a smaller field, and performance compresses upward, with the lowest score rising to 0.02 and the average improving to 0.59 (Table 2). This pattern is consistent with maturation effects (pipelines refined during DEV) and selection effects (fewer weak entries), producing stronger mid-pack outcomes.

At the top end, the best Pass@1 is slightly lower (0.95 vs. 0.99 in DEV), which is expected when feedback is constrained. The small top-line drop, paired with a higher mean, suggests that TEST emphasizes robustness over oppor-

Rank	Team Name	System Paper	Best Model	Pass@1
1	NALA_MAINZ	(Saadi et al., 2025)	GPT-5	0.95
2	Retriv	(Asib et al., 2025)	Qwen2.5-Coder-14B	0.93
3	Musafir	(Hasan et al., 2025)	Qwen2.5-14B-Instruct	0.92
4	AdversaryAI	(Riyad and Junaed, 2025)	Gemini 2.5 Pro	0.85
6	Code_Gen	(Agarwala et al., 2025)	GPT-5	0.84
7	TeamB2B	(Dihan et al., 2025)	Gemini-2.5-Pro	0.84
8	NSU_PiedPiper	(Fahmid et al., 2025)	Qwen2.5-Coder-14B	0.83
11	Barrier Breakers	(Jalil et al., 2025)	GPT OSS 120B	0.82
12	PyBhasha	(Dewan and Rifat, 2025)	Ensemble	0.80
13	JU_NLP	(Pal and Das, 2025)	GPT-4.1	0.77
16	AlphaBorno	(Rahman et al., 2025)	GPT-4o	0.72
17	PyBangla	(Islam et al., 2025)	Qwen3-8B	0.72
21	CUET_Expelliarmus	(Shahrier et al., 2025)	GPT-20B OSS	0.37
22	CodeAnubad	(Roy, 2025)	Gemma-2-9b-it	0.37
26	Troopers	(Farazi and Reza, 2025)	TigerLLM (RSFT)	0.32

Table 3: TEST phase results for the teams who submitted system description papers, ranked by Pass@1 scores (descending), scores rounded to two decimals. Complete results in Table 5 (Appendix B).

tunistic tuning: ceiling systems lose limited headroom, while the median gains from designs that encode safer defaults and broader guardrails. In effect, TEST acts as an external-validity probe, rewarding solutions that transfer beyond DEV's fully visible conditions and revealing residual brittleness in pipelines that depend on extensive test exposure (Table 2).

4 Results

DEV (63 teams). Table 4 (Appendix A) shows a clear separation between a small group of top systems and a wide middle. Under full test visibility, teams diagnose errors, adjust prompts and post-processing, and move up steadily. Top scores sit near perfect, but many teams still leave points on the table because of edge cases and inconsistent handling of problem specs. When scores are the same at the reported precision, we break ties by *shorter average solution length* (shorter wins). This favors solutions that meet the spec with minimal code rather than long, brittle fixes.

TEST (32 teams). Table 3 reflects how systems behave with less feedback. The middle of the leaderboard strengthens, while the very top tightens—high performers keep most of their lead, but not all of it. Designs that rely on stable defaults, careful I/O handling, and simple control flow hold up best; runs that depend on DEV-style trial-and-error drop back. We use the same tie-breaking rule here: *shorter average solution length* wins ties. In practice, this pushes teams toward concise, robust code that generalizes beyond the DEV environment.

5 Approaches

We briefly discuss the approaches of the 15 submitted *System* description papers in this section.

NALA_MAINZ (Saadi et al., 2025) (Rank 1)

The authors present the top-ranked system for the task. A lean multi-agent pipeline couples a code-generation agent with a selective debugger. The coder emits an initial solution and immediately runs unit tests; failures condense into error traces that guide the debugger to propose minimal, localized patches within a small step budget. The system augments supervision with matched external tests and lightweight auto-generated assertions, and it optionally translates Bangla prompts to English. Ablations indicate most gains come from error-trace–guided repair, with test augmentation adding complementary improvements.

Retriv (Asib et al., 2025) (Rank 2)

The authors propose a test-driven, feedback-guided framework. Their system uses a Qwen2.5-14B model (Hui et al., 2024b), fine-tuned with QLoRA (Dettmers et al., 2023), to generate an initial Python solution from translated English instructions. The code is immediately executed against unit tests. If a failure occurs, the error trace is fed back into the model prompt to guide a correction. This refinement loop is repeated up to three times with increasing temperature to encourage diverse solutions. The combination of parameter-efficient fine-tuning and iterative, execution-guided self-correction proved highly effective, securing the second-place rank.

Musafir (Hasan et al., 2025) (Rank 3)

This team employs a two-stage cascade pipeline. First, Bangla instructions are translated into English using a model optimized to preserve technical semantics. This step allows them to leverage powerful, English-centric code generation models. The translated prompt is then fed to a Qwen-based code generation model (Yang et al., 2024), which performs zero-shot code generation. The final output is validated using the provided unit tests. This direct translation-generation strategy effectively bridges the resource gap for Bangla, demonstrating a robust and high-performing approach that achieved third place in the competition.

AdversaryAI (Riyad and Junaed, 2025) (Rank 4)

The authors introduce TriGen (Think, Refine, and Generate), a system centered on a self-refinement loop. For open-source models, they use LoRA (Hu et al., 2022) to fine-tune on a dataset augmented with Chain-of-Thought (Wei et al., 2022) reasoning steps. The core of the system is an iterative process: an initial code solution is generated and tested. If it fails, the model receives the error feedback and is prompted to debug and correct its own output. This execution-guided refinement is applied to both their fine-tuned models and to a few-shot prompted Gemini 2.5 Pro, which yielded their top-performing submission.

Code_Gen (Agarwala et al., 2025) (Rank 6)

This work focuses on the impact of input quality, using a pipeline of preprocessing, translation, and assertion-based prompting with GPT-5. The authors first normalize the raw Bangla instructions to remove noise. Next, they translate the cleaned instructions to English to align with the model's strengths. Critically, they append the provided unit test assertions directly to the final prompt. This gives the model explicit examples of the required input-output behavior. Their experiments show that this assertion-augmented, translation-based approach significantly boosts performance, highlighting the importance of prompt clarity and context.

TeamB2B (Dihan et al., 2025) (Rank 7)

This team presents BanglaForge, a framework built on a retrieval-augmented (Lewis et al., 2020), dual-model collaborative pipeline. The system first uses TF-IDF to retrieve relevant solved examples, which are used for few-shot prompting. An initial "Coder" LLM generates a code solution. This solution is then passed to a "Reviewer" LLM, which validates the code, enhances its robustness, and refines it based on execution feedback from unit tests. This iterative cycle between the generator and reviewer agents, grounded by retrieved examples, effectively improves the final code's quality and correctness.

NSU_PiedPiper (Fahmid et al., 2025) (Rank 8)

The authors combine Chain-of-Thought (CoT) prompting (Zhou et al., 2024) with an iterative debugging loop (Liu et al., 2024). Using a Qwen-based model (Qwen Team et al., 2024), an initial solution is generated from a CoT prompt that encourages step-by-step reasoning. This code is then validated against unit tests. If any tests fail, the generated code and the resulting error messages are passed to a specialized debugger prompt. The model then attempts to fix the identified issues. This refinement process can be repeated up to three times, effectively using execution feedback to systematically correct errors from the initial reasoning phase.

Barrier Breakers (Jalil et al., 2025) (Rank 11)

This team introduces a novel approach that combines Test-Driven Development (TDD) and a Code Interpreter (CI) (Wang et al., 2024) without requiring model fine-tuning. First, in the TDD phase, the LLM generates additional test cases from the Bangla prompt. These new tests, combined with the provided one, are injected into the final prompt for code generation. In the CI phase, the generated code is executed in a sandbox. If any compilation or assertion errors occur, the error message is fed back to the model for up to five retry attempts, enabling iterative self-correction.

PyBhasha (Dewan and Rifat, 2025) (Rank 12)

The authors investigate the impact of instruction quality and model ensembling. They compare three instruction variants: original Bangla, English translations via Facebook NLLB (Team et al., 2022), and semantic-aware English rewrites using GPT-4.1. Finding the GPT-4.1 rewrites most effective, they implement a two-stage ensemble for their final submission. The primary model (Qwen2.5-Coder-14B) generates the initial solution. If this solution fails unit tests, it is passed to a secondary model (Claude Sonnet 4) as a fallback, leveraging the complementary strengths of different architectures to improve the overall success rate.

JU_NLP (Pal and Das, 2025) (Rank 13)

This team employs a straightforward yet effective zero-shot prompting strategy (Brown et al., 2020). They construct a detailed prompt that instructs the model to act as a senior Python developer, providing it with the original Bangla problem statement, the required function signature, and the visible unit tests. The prompt explicitly tells the model it can translate the instruction internally before generating the code. They test this approach across several proprietary models, with their best result coming from GPT-4.1, demonstrating the strong out-of-the-box, cross-lingual reasoning capabilities of modern frontier models.

AlphaBorno (Rahman et al., 2025) (Rank 16)

This work systematically evaluates several prompting strategies. After translating Bangla instructions to English with GPT-4o, they compare zero-shot, few-shot, and Chain-of-Thought baselines. Their key finding is that providing explicit behavioral constraints is more effective than abstract reasoning. Their best-performing method augments a zero-shot prompt with synthetic unit tests to cover edge cases. This is combined with a self-repair loop where failed execution feedback is used to prompt the model for a correction, with GPT-4o achieving the highest score under this configuration.

PyBangla (Islam et al., 2025) (Rank 17)

The authors introduce BanglaCodeAct, an agent-based framework inspired by the ReAct paradigm. Their system uses a general-purpose multilingual LLM (Qwen3-8B) in an iterative Thought-Code-Observation loop without any task-specific fine-tuning. For each problem, the agent first generates a ’Thought’ in Bangla outlining its plan. It then produces Python ’Code’ to implement the plan. This code is executed, and the ’Observation’ (output or error) is fed back to the agent. This cycle of self-correction continues until the code passes all unit tests, proving effective for low-resource code generation.

CUET_Expelliarmus (Shahrier et al., 2025) (Rank 21)

This team proposes a two-stage pipeline using the open-source GPT-20B OSS model. In the first stage, the Bangla instruction is translated to English and then refined using a one-shot prompt to create a well-structured specification. This refined English instruction is then passed to the second stage for code generation using a zero-shot prompt. The generated code is validated against unit tests. If a test fails, the traceback error is used as feedback to re-prompt the model, with this iterative correction loop running for up to five attempts.

CodeAnubad (Roy, 2025) (Rank 22)

This work tackles the extreme data scarcity of the task with an iterative self-improvement strategy. The authors first fine-tune a Gemma-2-9b model on the initial 74 training samples using QLoRA (Dettmers et al., 2023). This model is then used to generate solutions for the development set. All solutions that pass the unit tests are harvested and added to the training set. The model is then re-trained on this augmented dataset. This process creates a positive feedback loop, progressively improving performance by curating a high-quality, in-domain dataset from the model's own verified outputs.

Troopers (Farazi and Reza, 2025) (Rank 26)

The authors implement a reward-selective fine-tuning (RSFT) pipeline (Dong and others, 2023). The process begins by sampling multiple candidate programs from a base model for each Bangla prompt. Each candidate is executed in a sandbox, and only those that pass all unit tests (the "winners") are retained. This curated set of high-quality, execution-verified instruction-code pairs forms the dataset for supervised fine-tuning (SFT). The base model is then efficiently updated on this dataset using LoRA adapters, selectively reinforcing correct program synthesis without complex reinforcement learning.

6 Analysis

Since the task focuses on generation, all the systems are built around one or more LLMs. Table 6 (Appendix C) lists all the models used by each system.

6.1 Preference on LLMs

Participants have used a total of 20 different LLMs, including 6 proprietary and 14 open-source models. As Figure 3 illustrates that TigerLLM is the most used model along with two other open-source ones (LLaMA 3 and Qwen2.5).

6.2 Best Performing LLMs

As Figure 4 shows, Qwen2.5 was the best-performing model by most systems, followed by the proprietary models and some other open-source models.

6.3 Methodologies

As shown in Figure 5, teams build upon a common foundation. Prompting is nearly universal, while a majority (8 of 15 teams) use Machine Translation to leverage powerful English-centric models, a key strategy for systems like Musafir (R3). Five teams employ Finetuning to specialize models; Retriv (R2) uses QLORA for efficiency, while Troopers (R26) implements a reward-selective pipeline (RSFT) to train on verified-correct code.

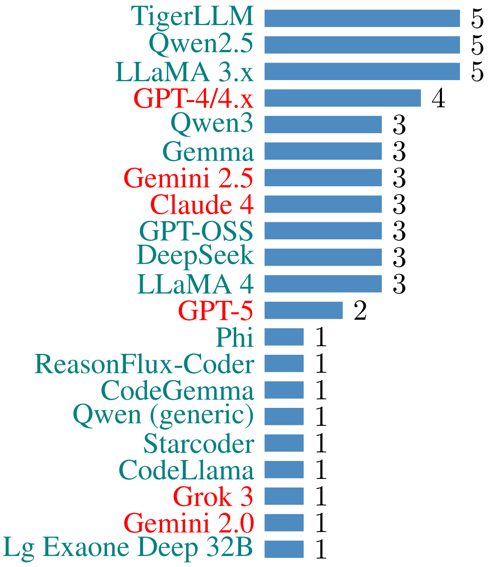


Figure 3: Most used LLMs by the submitted systems. Proprietary models are in red.

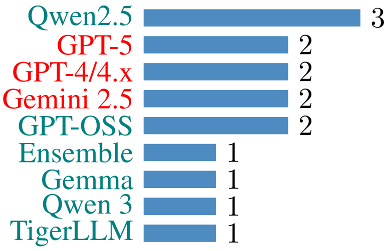


Figure 4: Best performing LLMs by the submitted systems. Proprietary models are in red.

The most critical differentiator for top-tier systems is the implementation of a self-correcting feedback loop. This is often initiated with Chain-of-Thought (CoT) prompting to improve the model's initial reasoning, as seen with NSU_PiedPiper (R8). The core of this approach is Iterative Self-Correction, where generated code is executed and any resulting errors are fed back to the model for debugging. This refinement process proves central to the success of the highest-performing teams, including NALA_MAINZ (R1), Retriv (R2), and AdversaryAI (R4).

A few teams explore more specialized strategies. TeamB2B (R7) utilizes Retrieval-Augmented Generation (RAG) to provide models with relevant examples, while Barrier Break-

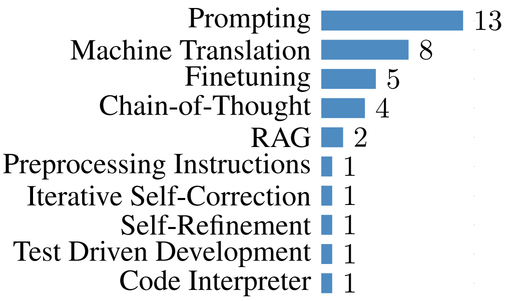


Figure 5: Most used methodologies by the submitted systems.

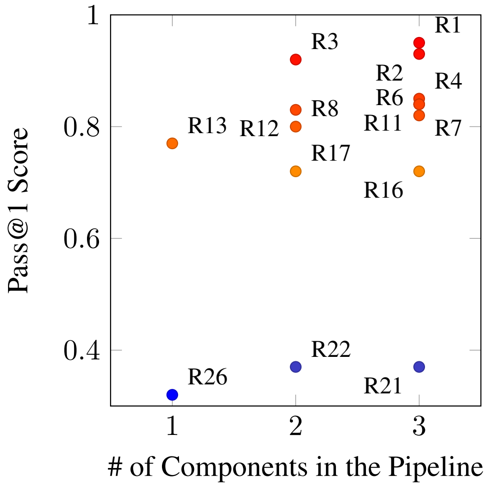


Figure 6: Correlation between pipeline complexity (number of distinct high-level components/methodologies employed) and the final Pass@1 score for the 15 teams with system description papers. Each point is labeled with the team's final rank.

ers (R11) uniquely combines Test-Driven Development (TDD) with a Code Interpreter (CI) for safe, iterative refinement. These advanced methods underscore a clear trend: top performance requires moving beyond foundational techniques to build robust, multi-step systems that emulate real-world development workflows.

6.4 Pipeline Components

Figure 6 visualizes the relationship between the architectural complexity of a system and its final score. A strong positive correlation is evident: systems employing a greater number of integrated methodologies consistently achieved higher performance. Notably, five of the top eight teams utilized complex pipelines integrating at least three distinct techniques, such as translation, Chain-of-Thought, and iterative self-correction. This trend highlights that success in this task was not merely dependent on model choice, but was significantly driven by the sophistication of the overall pipeline. Simpler approaches, while effective to a degree, generally did not reach the top performance tiers.

7 Conclusion

In this shared task, the first of its kind for Bangla code generation, we successfully benchmarked the capabilities of modern LLMs on a low-resource language. We observed a clear methodological trend from the diverse systems submitted: top performance was not driven by model choice alone, but by pipeline complexity. We found that the most effective systems implemented robust, multi-step workflows with self-correction loops that emulate a developer's iterative debugging process. We observe that, finetuning and machine translation were the most effective methods during the test phase. While open-source models only performed better after pairing them up with some test-driven coding tools.

Our results establish a strong baseline and highlight the effectiveness of agentic, self-refining architectures. For future work, we recommend focusing on developing capable Bangla-native code models to reduce the dependency on translation, expanding benchmark complexity, and exploring how these successful pipeline strategies can be transferred to other languages.

We plan to build on our findings, and our priorities include refining these agentic workflows, developing native Bangla code models to reduce the current dependency on translation, and increasing benchmark complexity to repository-level tasks like bug fixing. Advancing these areas will not only improve Bangla code generation but also provide a transferable blueprint for other under-resourced languages, making AI-driven software development more globally accessible.

Limitations

While our task is intentionally focused on generating self-contained, function-level Python code, we acknowledge this does not encompass the full complexity of real-world software engineering. This focused scope, however, was a deliberate design choice to establish a clear, controlled, and reproducible benchmark—a critical first step for a new task in a low-resource language. Similarly, our use of the stringent Pass@1 metric, which is standard in code generation benchmarks, provides an unambiguous signal of functional correctness. While many top systems relied on translating prompts to English, we view this not as a limitation of the task, but as a key finding that accurately reflects the current state-of-the-art strategies for bridging the resource gap, providing a realistic baseline for future work to improve upon.

Ethical Considerations

The datasets used in this task are derived from publicly available, open-source benchmarks, mitigating data privacy concerns. A primary goal of our work is to enhance the accessibility of programming tools for Bangla speakers, promoting linguistic inclusivity in technology. However, we acknowledge that any code generation system carries a potential risk of misuse for generating malicious code, although the function-level scope of our task makes this risk indirect. The prevalent use of proprietary models also means we rely on the safety and bias mitigations implemented by model providers. While the technical nature of the prompts limits the potential for social bias, the common strategy of translating prompts to English could amplify biases present in the target English-centric LLMs.

References

Abhishek Agarwala, Shifat Islam, and Emon Ghosh. 2025. Code_gen at blp-2025 task 2: Banglacode: A crosslingual benchmark for code generation with translation and assertion strategies. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Kabir Ahuja, Anirudh Das, Sandipan Das, Ashwini Deshpande, Sebastian Gehrmann, Anup Gopinath, Arya Guha, Pooja Kumar-Jois, Prem Mani, Ashwin Paranjape, et al. 2023. Mega: Multilingual evaluation of generative ai. *arXiv preprint arXiv:2310.10567*.

K M Nafi Asib, Sourav Saha, and Mohammed Moshiul Hoque. 2025. Retriv at blp-2025 task 2: Test-driven feedback-guided framework for bangla-to-python code generation. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. In *arXiv preprint arXiv:2108.07732*.

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*.

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*.

Damián Blasi, Antonios Anastasopoulos, and Graham Neubig. 2022. Systematic inequalities in language technology performance across the world's languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. In *arXiv preprint arXiv:2107.03374*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Foyez Ahmed Dewan and Nahid Montasir Rifat. 2025. Pybhasha at blp-2025 task 2: Effectiveness of semantic-aware translation and ensembling in bangla code generation. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Mahir Labib Dihan, Sadif Ahmed, and Md Nafiu Rahman. 2025. Teamb2b at blp-2025 task 2: Banglaforge: Llm collaboration with self-refinement for bangla code generation. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Yao Dong et al. 2023. Raft: Reward ranked finetuning for generative foundation models. In *NeurIPS 2023*.

Syed Mohammed Sartaj Ekram, Adham Arik Rahman, Md Sajid Altaf, Mohammed Saidul Islam, Tareq Mahmood Jamil, Shadman Sakib Alam, Irfan Kabir, Mohammad Nasim, Enamul Hossain, and Nawshad Akhter. 2022. Banglarqa: A benchmark dataset for under-resourced bangla language reading comprehension-based question answering with diverse question-answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Ahmad Fahmid, Fahim Foysal, Wasif Haider, Shafin Rahman, and Md Adnan Arefeen. 2025. Nsu_piedpiper at blp-2025 task 2: A chain-of-thought with iterative debugging approach for code generation with bangla instruction. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Musa Tur Farazi and Nufayer Jahan Reza. 2025. Troopers at blp-2025 task 2: Reward-selective fine-tuning based code generation approach for bangla prompts. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Sakibul Hasan, Md Tasin Abdullah, Abdullah Al Mahmud, and Ayesha Banu. 2025. Musafir at blp-2025 task 2: Generating python code from bangla prompts using a multi-model cascade and unit test validation. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024a. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024b. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Jahidul Islam, Md Ataullha, and Saiful Azad. 2025. Pybangla at blp-2025 task 2: Enhancing bangla-to-python code generation with iterative self-correction and multilingual agents. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Sajed Jalil, Shuvo Saha, and Hossain Mohammad Seym. 2025. Barrier breakers at blp-2025 task 2: Enhancing bengali llm code generation capabilities through test driven development and code interpreter. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Myle Ott, Wen-tau Chen, Alexis Conneau, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

M Liu et al. 2024. Iterative debugging for neural code generation. *ACM Transactions on Programming Languages and Systems*, 46(3):1–33.

Pritam Pal and Dipankar Das. 2025. Ju_nlp at blp-2025 task 2: Leveraging zero-shot prompting for bangla natural language to python code generation. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Liliana Pasquale, Antonino Sabetta, Marcelo d'Amorim, Péter Hegedűs, Mehdi Tarrit Mirakhorli, Hamed Okhravi, Mathias Payer, Awais Rashid, Joanna CS Santos, Jonathan M Spring, et al. 2025. Challenges to using large language models in code generation and repair. *IEEE Security & Privacy*, 23(2):81–88.

Qwen Team, Jinze Bai, Shuai Xu, Yankai Zhang, Zhenru Wang, Songyang Wang, Ziyan Li, Wang Wang, Li Wang, Siyuan Liu, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Mohammad Ashfaq Ur Rahman, Muhtasim Ibteda Shochcho, and Md Fahim. 2025. Alphaborno at blp-2025 task 2: Code generation with structured prompts and execution feedback. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Christian Newman, and Marcos Zampieri. 2024. Code llms: A taxonomy-based survey. In *Proceedings of IEEE BigData*.

Nishat Raihan, Mohammed Latif Siddiq, Joanna CS Santos, and Marcos Zampieri. 2025c. Large language models in computer science education: A systematic literature review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 938–944.

Omar Faruqe Riyad and Jahedul Alam Junaed. 2025. Adversaryai at blp-2025 task 2: A think, refine, and generate (trigen) system with lora and self-refinement for code generation. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Soumyajit Roy. 2025. Codeanubad at blp-2025 task 2: Efficient bangla-to-python code generation via iterative lora fine-tuning of gemma-2. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Hossain Shaikh Saadi, Faria Alam, Mario Sanz-Guerrero, Minh Duc Bui, Manuel Mager, and Katharina von der Wense. 2025. Nala_mainz at blp-2025 task 2: A multi-agent approach for bengali instruction to python code generation. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Sheikh Shafayat, H Hasan, Minhajur Mahim, Rifki Putri, James Thorne, and Alice Oh. 2024. BEnQA: A question answering benchmark for Bengali and English. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Md Kaf Shahrier, Suhana Binta Rashid, Hasan Mesbaul Ali Taher, and Mohammed Moshiul Hoque. 2025. Cuet_expelliarmus at blp2025 task 2: Leveraging instruction translation and refinement for bangla-to-python code generation with open-source llms. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Asian Chapter of Association for Computational Linguistics (AACL).

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

NLLB Team, Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Guillaume Wenzek, Kevin Lin, Tatiana Tran, Shruti Le, et al. 2022. No Language Left Behind: scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Md Nafis Uddin, Masum Khan, Nabila Hasan, and Mahmudul Hossain. 2023. Exploring code-mixed bangla text in large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Tianyi Wang, Yang Ye, Panupong Pasupat, Aohan Wan, Grant Friedman, Jiacheng Tu, Maya Schaar, Jason Wei, Suriya Gunasekar, Matthew Richardson, et al. 2023. Babelcode: Llm as a polyglot programmer. *arXiv preprint arXiv:2303.03845*.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. *arXiv preprint arXiv:2402.04391*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Abdullah Khan Zehady, Safi Al Mamun, Naymul Islam, and Santu Karmaker. 2024. Bongllama: Llama for bangla language. *arXiv preprint arXiv:2410.21200*.

Denny Zhou et al. 2024. Advancing chain-of-thought reasoning in large language models. *Nature Machine Intelligence*, 6(1):45–58.

A DEV Phase Results

Rank	Team_Name	Score
1	BRACU_CL	1.00
2	NALA_MANIZ_2	1.00
3	Team_Trinity	0.99
4	not_Decided	0.99
5	Code_Gen	0.97
6	Team B2B	0.97
7	Musafir	0.94
8	Oleksandr Usyk	0.94
9	PyBangla	0.94
10	alubhorta	0.94
11	Metaphor	0.93
12	NO Name	0.91
13	Alpha Borno	0.90
14	Gradient Masters	0.88
15	NLPirates	0.87
16	PyBhasha	0.86
17	Nsu_PiedPiper	0.85
18	CUET_SIURS	0.84
19	JU_NLP	0.84
20	Retriv	0.84
21	SamNLP	0.84
22	fallen_dark-358115	0.84
23	Ecstasy	0.83
24	NeuralCoders	0.83
25	CUET_DuoBingo	0.82
26	3_idiots	0.78
27	BanglaBytes	0.73
28	Py_Chunker	0.68
29	delayed	0.64
30	BarrierBreakers	0.62
31	AdversaryAI	0.60
32	Team_AA	0.56
33	BLPCG	0.51
34	theDarkKnights	0.48
35	soumyajit	0.47
36	wspr	0.46
37	rms92	0.44
38	NeuralCoders	0.41
39	KodomAli Coders	0.38
40	PrompterXPrompter	0.38
41	UIU_NLP	0.32
42	unknown	0.31
43	Md_Abdur_Rahman	0.29
44	CUET_Zahra_Duo	0.17
45	Wahid	0.11
46	Organizers	0.10
47	Quasar	0.10
48	Team_Ban	0.10
49	turtur	0.10
50	SoloGuy	0.09
51	huday	0.09
52	troublemaker	0.09
53	Arekta Team	0.08
54	Kaf	0.08
55	None	0.08
56	Sweet Dreams	0.08
57	disco	0.08
58	cuet_1376	0.03
59	tryNLP	0.03
60	nafiurahman-353732	0.01
61	CUET_NLP_Zahra_Duo	0.00
62	Troopers	0.00
63	programophile	0.00

Table 4: DEV phase results. 63 Teams - ranked by Pass@1 scores (descending) — scores rounded to two decimals.

B TEST Phase Results

Rank	Team Name	Pass@1
1	NALA_MAINZ	0.95
2	Retriv	0.93
3	Musafir	0.92
4	AdversaryAI	0.85
5	BRACU_CL	0.84
6	Code_Gen	0.84
7	TeamB2B	0.84
8	NSU_PiedPiper	0.83
9	One Braincell	0.83
10	fallen_dark-370156	0.83
11	Barrier Breakers	0.82
12	PyBhasha	0.80
13	JU_NLP	0.77
14	This Team has no name	0.77
15	NLPirates	0.74
16	AlphaBorno	0.72
17	PyBangla	0.72
18	CUET_DuoBingo	0.70
19	CUET_SIURS	0.67
20	Ecstasy	0.66
21	CUET_Expelliarmus	0.37
22	CodeAnubad	0.37
23	Gradient Masters	0.36
24	team_trinity	0.36
25	nidala	0.33
26	Troopers	0.32
27	Team Random	0.28
28	delayed	0.18
29	Organizers	0.17
30	huday	0.09
31	SyntaxMind	0.08
32	Team Random	0.02

Table 5: TEST phase results for the teams who submitted system description papers. 32 Teams — ranked by Pass@1 scores (descending), scores rounded to two decimals.

C LLMs Used by Teams

The following table details the various Large Language Models (LLMs) employed by the teams who submitted system papers, sorted by their final rank in the competition.

Rank	Team Name	Models Used
1	NALA_MAINZ	GPT 5, Claude 4, Gemini 2.5
2	Retriv	Phi, Qwen3, Qwen2.5-Coder, Llama-3.1, ReasonFlux-Coder, codegemma
3	Musafir	Qwen
4	AdversaryAI	TigerLLM, Gemma 3, Gemini 2.5, Llama3, Qwen3, Qwen2.5
6	Code_Gen	GPT 4, GPT 5, LLaMA 4, TigerLLM, Deepseek
7	TeamB2B	Gemini 2.5, Gemma-1B, GPT-OSS, DeepSeek-R1, Gemini2.0, Lg Exaone Deep
8	NSU_PiedPiper	Qwen2.5-Coder-14B
11	Barrier Breakers	LLaMA 4, Llama 3.2, GPT-OSS
12	PyBhasha	GPT 4, Claude 4, TigerLLM-9B, Qwen2.5-Coder, LLaMA-3.1
13	JU_NLP	GPT 4, LLaMA 4
16	AlphaBorno	GPT 4, Claude 3.7 Sonnet, Qwen Coder 2.5, Grok 3
17	PyBangla	TigerLLM, Qwen3, Llama-3.1, DeepSeek-Coder-V2
21	CUET_Expelliarmus	GPT-20B-OSS
22	CodeAnubad	Gemma, Starcoder, CodeLlama
26	Troopers	TigerLLM

Table 6: All the LLMs used by the teams (only includes the submitted system description papers).

Overview of BLP-2025 Task 1: Bangla Hate Speech Identification

Md Arid Hasan[1], **Firoj Alam**[2], **Md Fahad Hossain**[3], **Usman Naseem**[4],
Syed Ishtiaque Ahmed[1]
[1]University of Toronto, Canada, [2]Qatar Computing Research Institute, Qatar
[3]Daffodil International University, [4]Macquarie University, Australia
{arid, ishtiaque}@cs.toronto.edu, fialam@hbku.edu.qa
https://multihate.github.io

Abstract

Online discourse in Bangla is rife with nuanced toxicity expressed through code-mixing, dialectal variation, and euphemism. Effective moderation thus requires fine-grained detection of hate **type**, **target**, and **severity**, rather than a binary label. To address this, we organized the Bangla Hate Speech Identification Shared Task at **BLP 2025 workshop**, co-located with **IJCNLP-AACL 2025**, comprising three subtasks: (1A) hate-type detection, (1B) hate-target detection, and (1C) joint prediction of type, target, and severity in a multi-task setup. The subtasks attracted 161, 103, and 90 participants, with 36, 23, and 20 final submissions, respectively, while a total of 20 teams submitted system description papers. The submitted systems employed a wide range of approaches, ranging from classical machine learning to fine-tuned pretrained models and zero-/few-shot LLMs. We describe the task setup, datasets, and evaluation framework, and summarize participant systems. All datasets and evaluation scripts are publicly released.[1]

1 Introduction

Hate speech detection has emerged with the wide use of social media and online communication platforms, where users can rapidly share opinions, comments, and multimedia content. The proliferation of such content has led to an increase in harmful content (Walther, 2022). This has also facilitated the spread of hate speech language that targets individuals or groups based on attributes such as religion, ethnicity, gender, or political affiliation (Fortuna and Nunes, 2018). Therefore, detecting hate speech automatically is crucial for maintaining safe online environments and preventing real-world consequences such as discrimination and violence. While substantial progress has been made for English (Albladi et al., 2025) and other high-resource languages (Das et al., 2024), hate speech detection in Bangla remains a significant challenge due to the scarcity of annotated datasets and linguistic diversity (Sharma et al., 2025; Das et al., 2022b; Haider et al., 2025; Romim et al., 2022).

Early studies in Bangla hate speech detection focused on classical machine learning methods (Kiela et al., 2020; Mridha et al., 2021; Romim et al., 2022), deep learning models (Romim et al., 2022; Keya et al., 2023), and pretrained models that are primarily designed for English (Mridha et al., 2021). However, these methods struggle in understanding the deep cultural, social, and linguistic nuances that shape hate expression in Bangla (Al Maruf et al., 2024). These include context-sensitive slurs, metaphorical or sarcastic insults, and region-specific idiomatic phrases that are often misinterpreted or overlooked by standard models (Jahan et al., 2022). Recently, large language models (LLMs) such as GPT-5, Gemini (Comanici et al., 2025), Qwen (Yang et al., 2025), and Llama (Dubey et al., 2024) have achieved impressive generalization in NLP tasks but often underperform in hate speech detection for low-resource languages like Bangla. Their limited ability to grasp cultural nuances, implicit hate, and context-specific expressions (Zahid et al., 2025) underscores the need for domain adaptation and culturally aware training strategies.

Prior studies are limited to single-task datasets, focusing on only one dimension, such as hate type, which restricts the development of more sophisticated models capable of performing a multi-faceted analysis (e.g., simultaneously identifying a comment's type, severity, and target). Therefore, we emphasize community engagement and organized a shared task at BLP 2025[2] to address this challenge. The task consists of three subtasks[3] and

[1] https://github.com/AridHasan/blp25_task1

[2] https://blp-workshop.github.io/

[3] Subtask 1A: Identifying type of hate, Subtask 1B: Identifying target of hate, Subtask 1C: Identifying type, severity,

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 388–401
December 23, 2025 ©2025 Association for Computational Linguistics

aims to foster the development of robust, multi-task Bangla hate speech detection systems by providing a large, carefully curated dataset spanning multiple domains. By encouraging diverse approaches ranging from classical models to LLMs and solutions, the shared task promotes collaboration and the exploration of various techniques tailored to the cultural, social, and linguistic nuances.

A total of 161, 103, and 89 teams registered for subtasks 1A, 1B, and 1C, respectively, with 37, 24, and 21 teams making official submissions on the test set. Among these, 19 teams also submitted system description papers. On the development-test set, there were 925 submissions from 56 teams for 1A, 312 submissions from 23 teams for 1B, and 224 submissions from 18 teams for 1C. For the final test set, the number of submissions were 421, 271, and 202 for subtasks 1A, 1B, and 1C, respectively.

2 Task and Dataset

2.1 Task

This shared task focused on detecting hate speech in Bangla, a low-resource language with rich morphology and regional dialects. To comprehensively address the complexity of hateful content, we divided the task into three subtasks: *(i)* identifying the *Type of Hate*, *(ii)* determining the *Target of Hate*, and *(iii)* a comprehensive multi-task setup that jointly predicts the *Type of Hate*, *Severity of Hate*, and *Target of Hate*. This design aimed to move beyond simple binary detection and encourage the development of models capable of addressing the nuanced linguistic and social dimensions of hate speech in Bangla. Task descriptions are provided below.

Subtask 1A: Type of Hate This subtask is defined as *"detect the type of hate that is expressed in the text"*. This is a multi-class classification task that requires determining whether a given instance falls into one of the following categories: *Abusive*, *Sexism*, *Religious Hate*, *Political Hate*, *Profane*, or *None*.

Subtask 1B: Target of Hate This subtask is defined as *"detect the target of hate that is expressed towards **whom** in the text"*. It is formulated as a multi-class classification problem, where the goal is to determine whether the hateful expression is aimed at *Individuals*, *Organizations*, *Communities*, *Society*, or *None*.

Subtask 1C: Multi-task Setup This subtask is designed as a multi-task learning setup that jointly addresses three dimensions of hate speech detection in Bangla: *Type of Hate*, *Severity of Hate*, and *Target of Hate*. Unlike the single-task formulations in Subtask 1A and Subtask 1B, this setup requires models to perform simultaneous predictions across all three aspects for each input text. The motivation behind this design is to encourage the development of models capable of capturing the inter-dependencies between different facets of hateful content, such as the way severity may vary depending on the type of hate or how certain targets are more likely to be associated with specific hate categories. Framing the problem as a multi-task learning challenge encourages the development of more robust and context-sensitive systems that go beyond individual classification tasks and better capture the complex, multidimensional characteristics of hate speech in Bangla. The target classes for *Type of Hate* and *Target of Hate* are the same as defined in Subtask 1A and Subtask 1B, respectively, while the *Severity of Hate* task categorizes instances into *Severe*, *Mild*, and *Little to None*. Here, *Severe* denotes strongly derogatory or inciteful content, *Mild* refers to moderately offensive or implicitly hateful expressions, and *Little to None* indicates content with minimal or no hateful intent.

2.2 Dataset

We utilized the *BanglaMultiHate* dataset (Hasan et al., 2025b) for this shared task. This dataset comprises comments from YouTube, sourced from the Somoy Bangla News channel.[4] This dataset covers 19 different topics, such as *Disaster, Entertainment, Health, Politics, Religion, Science, Sports*, etc. For this shared task, we utilize the training set as the official training set of the shared task. The development set was further divided into development and development-test[5] subsets using a stratified sampling approach to preserve class balance. Finally, the test is used for system evaluation and participant ranking. The detailed distribution of the data split is presented in Table 1.

and target of hate in multi-task setup

[4] https://www.youtube.com/@somoynews360

[5] This development test set is used as a test set for the development phase.

Class	Train	Dev	DT	Test	Total
Type of Hate					
Abusive	8,212	564	549	2,312	**11,637**
Political Hate	4,227	291	283	1,220	**6,021**
Profane	2,331	157	185	709	**3,382**
Religious Hate	676	38	40	179	**933**
Sexism	122	11	8	29	**170**
None	19,954	1,451	1,447	5,751	**28,603**
Total	**35,522**	**2,512**	**2,512**	**10,200**	**50,746**
Severity of Hate					
Litle to None	23,489	1,703	1,714	6,737	**33,643**
Mild	6,853	483	426	2,001	**9,763**
Severe	5,180	326	372	1,462	**7,340**
Total	**35,522**	**2,512**	**2,512**	**10,200**	**50,746**
Target of Hate					
Community	2,635	179	159	759	**3,732**
Individual	5,646	364	391	1,571	**7,972**
Organization	3,846	292	292	1,152	**5,582**
Society	2,205	141	142	625	**3,113**
None	21,190	1,536	1,528	6,093	**30,347**
Total	**35,522**	**2,512**	**2,512**	**10,200**	**50,746**

Table 1: Class label distribution of the shared task dataset. DT: development-test.

Annotation and Annotators Agreement The annotation of the *BanglaMultiHate* dataset (Hasan et al., 2025b) was conducted by a trained team of 35 native Bangla-speaking undergraduate students, with each comment labeled independently by three annotators and finalized by majority vote or consensus when needed. Quality checks and supervision ensured consistent standards. Inter-annotator agreement, measured using Fleiss' Kappa, showed substantial to almost perfect agreement across tasks, with scores of 0.71 for type of hate, 0.84 for severity of hate, and 0.79 for target of hate, while more fine-grained tasks yielded slightly lower agreement due to increased complexity.

3 Evaluation Framework

3.1 Evaluation Measures

We used the *unweighted Micro-F1 score* as the evaluation metric for Subtask 1A and 1B, while weighted Micro-F1 score is used for Subtask 1C, with the corresponding datasets and evaluation scripts made publicly accessible online.[6] To establish reference points, we included the majority and random baselines along with the n-gram. The majority baseline predicts the most frequent class in the training data for every instance in the test set, while the random baseline assigns classes to test instances uniformly at random. We also provide a simple n-gram ($n = 1$) baseline using 5,000 features, with a linear SVM implemented to capture surface-level lexical patterns.

3.2 Task Organization

For the shared task, we released four datasets: the training set, development set, development-test set, and test set for each subtask, as summarized in Table 1. The development set was intended for hyperparameter tuning, while the development-test set was provided without labels to enable participants to assess their systems during the development phase. The test set was utilized for the final evaluation and ranking of submissions. All the subtasks (Subtask 1A,[7] Subtask 1B,[8] and Subtask 1C[9]) of this shared task was conducted in two phases, with the submission platform hosted on CodaBench.

Development Phase During this phase, participants were provided with the training set, development set, and development-test set. The development-test set was released without gold labels to ensure fair competition. This design encouraged participants to iteratively refine and optimize their models using the labeled training and development sets, while evaluating their systems on the unlabeled development-test set. A live leaderboard was made available throughout this phase, allowing teams to monitor the relative performance of their submissions in real time and to benchmark their approaches against other participants. This competitive setup fostered active engagement and provided valuable insights into the effectiveness of different modeling strategies prior to the final evaluation stage.

Evaluation Phase In this phase, the test set was released without gold labels, and participants were allotted a eight-day window to submit their final predictions. The test set served as the basis for the official evaluation and ranking of systems. To preserve fairness and prevent overfitting to the test data, the leaderboard was kept private during this phase. While participants were permitted to submit multiple systems (per day 100 submissions and in total 1000 submissions), the corresponding evaluation scores were withheld from them. For the

[6] https://github.com/AridHasan/blp25_task1

[7] https://www.codabench.org/competitions/9559/

[8] https://www.codabench.org/competitions/9560/

[9] https://www.codabench.org/competitions/9561/

final ranking, only the last valid submission from each team was considered, ensuring consistency and comparability across participants.

The test set along with its gold labels was released after the competition concluded, allowing participants to perform additional experiments, conduct error analyses, and further refine their models.

4 Results and Overview of the Systems

4.1 Results

In this section, we present the outcomes of the shared task across both phases. Overall, participation was strong, with 56, 23, and 18 teams submitting systems during the development phase and 36, 23, and 18 teams in the evaluation phase for Subtask 1A, 1B, and 1C, respectively. Tables 2, 4, and 5 report the performance of all submitted systems on the development-test and test sets, alongside the majority and random baselines for comparison for Subtask 1A, 1B, and 1C, respectively. The official ranking was determined based on results from the test set. Notably, some teams participated only in the development phase but not in the evaluation phase, and vice versa, as indicated by ✗.

A comparison of results across the development-test and test sets indicates that performance differences among teams were minimal across three subtasks. This suggests that the models generally did not exhibit overfitting to the development-test set. In several instances, the systems even achieved higher performance on the test set than on the development-test set, highlighting the robustness and generalizability of the submitted approaches.

Table 3 provides a comprehensive overview of the approaches employed by participating teams across the three subtasks. The majority of teams relied on transformer-based architectures, particularly BanglaBERT, XLM-RoBERTa, and MuRIL, reflecting their effectiveness for Bangla and low-resource contexts. Several systems integrated ensemble strategies, combining multiple fine-tuned models to enhance robustness and performance. Moreover, we observed that systems employing ensemble techniques achieved the highest rankings on the leaderboard on all three subtasks. A smaller subset of teams experimented with classical machine learning and neural network baselines, often as complementary or comparative models. Additionally, data preprocessing and data augmentation were common practices to improve text quality and address class imbalance. A few teams further adopted LLMs (such as GPT-4.1, Llama3, Gemma 2, and Qwen3) and few-shot prompting approaches (e.g., Qwen-based systems), showcasing an emerging shift toward generative and low-resource adaptive methods.

4.2 Overview of the Systems

We summarize each participating system and its corresponding ranking on the leaderboard below.

Code_Gen (Islam et al., 2025) achieved the best performance in subtask 1A, ranked 2^{nd} and 3^{rd} for subtask 1B and subtask 1C, fine-tuned BanglaBERT (Bhattacharjee et al., 2021), multilingual E5 (Wang et al., 2024), MuRIL (Khanuja et al., 2021), XLM-RoBERTa (Conneau et al., 2020), and DistilBERT using token-aware adversarial contrastive training and layer-wise learning rate decay to enhance optimization and stability. Initially, the authors preprocessed through normalization, cleaning, and tokenization, and then incorporated data augmentation with a 70:30 train–validation split. Moreover, the authors utilized individual model logits that were generated and subsequently ensembled through different combinations of models to improve predictive performance.

SyntaxMind (Riad, 2025) integrates contextual language representations with sequential and local feature extraction mechanisms to enhance the classification task. To generate contextual embeddings, the authors used BanglaBERT encoder, which is then processed in parallel through a CNN that captures local n-gram patterns through multiple kernel sizes, and a GRU utilizes sequential dependencies with bidirectional recurrence. Moreover, both CNN and GRU employ self-attention, and the outputs of the CNN and GRU attentions are then fused through a dense layer. This team ranked 2^{nd} and 5^{th} in subtask 1A and 1B, respectively.

TeamHateMate (Hasan and Mahim, 2025) fine-tuned BanglaBERT using a two-stage cascading architecture for all three sub-tasks: a binary classifier to separate hate from non-hate, followed by a multi-class classifier for fine-grained categorization. Each stage was optimized through k-fold cross-validation and ensembled through majority voting. The authors also incorporated attention pooling, dropout, and hidden layers to enhance performance and tuned hyperparameters separately for each subtask. Their system ranked 4^{th} in subtask 1A, while ranked 1^{st} in both subtask 1B and 1C. The authors further attempted data augmentation via back translation and class balancing; however,

R.	Team Name	Dev P.	Eval. P.
1	Code_Gen (Islam et al., 2025)	0.7580	0.7362
2	SyntaxMind (Riad, 2025)	0.7440	0.7345
3	zannatul_007	0.7440	0.7340
4	TeamHateMate (Hasan and Mahim, 2025)	0.7544	0.7331
5	Ecstasy (Hasan et al., 2025a)	0.7564	0.7328
6	Gradient Masters (Rahman et al., 2025b)	0.7488	0.7323
7	Catalyst (Hasan and Hasan, 2025)	0.7572	0.7305
8	BElite (Tripty et al., 2025b)	0.7444	0.7282
9	Retriv (Saha et al., 2025)	0.7572	0.7275
10	CoU-CU-DSG (Alam et al., 2025)	0.7217	0.7273
11	CUET-NLP_Zenith (Hossan et al., 2025)	0.7357	0.7263
12	NSU_MILab (Rahman et al., 2025a)	0.7138	0.7250
13	abid_al_hossain	0.7416	0.7238
14	PentaML (Tahmid et al., 2025)	0.7162	0.7178
15	HateNetBN (Anam and Mazumder, 2025)	0.7365	0.7133
16	Computational StoryLab (Prama et al., 2025)	0.7404	0.7111
17	minjacodes9	0.5852	0.7075
18	Heisenberg (Yasir, 2025)	0.7086	0.7070
19	pritampal98	0.7373	0.7057
20	Bahash-AI (Laskar and Paul, 2025)	✗	0.7028
21	Velora (Sayem and Rahman, 2025)	0.7197	0.7013
22	fatin_anif	✗	0.6954
23	PerceptionLab (Fahim and Khan, 2025)	0.6584	0.6941
24	adriti12	0.3264	0.6921
25	nuralflow	0.7038	0.6901
26	Team_NSU_Strugglers	0.6899	0.6871
27	CUET_Sntx_Srfrs (Tripty et al., 2025a)	✗	0.6867
28	abir_bot69	0.6393	0.6840
29	antara_n_15	0.6899	0.6815
30	PromptGuard (Hossan and Roy Dipta, 2025)	0.6879	0.6761
31	quasar	0.1075	0.6733
32	shahriar_9472	0.6720	0.6689
33	intfloat	0.6712	0.6634
34	naim-parvez	✗	0.6587
35	Baseline (Majority)	0.5760	0.5638
36	teddymas	✗	0.4589
37	Baseline (Random)	0.1465	0.1638
38	mizba	0.7237	0.1077
–	messalmonem	0.7588	✗
–	nur_163	0.7568	✗
–	cuet_1376	0.7393	✗
–	manik	0.7361	✗
–	Tensoryus	0.7357	✗
–	phantom_troupe	0.7345	✗
–	hasnat	0.7329	✗
–	rashfi_21	0.7325	✗
–	Md.Fahad Ali	0.7313	✗
–	rabeya_akter	0.7237	✗
–	foysal_ahmed	0.7197	✗
–	md_abdur_rahman	0.7166	✗
–	no_name	0.7102	✗
–	saminyasir007	0.7062	✗
–	shuvodwip_saha	0.7030	✗
–	mhd88	0.6979	✗
–	tesnik	0.6883	✗
–	walisa_alam	0.6879	✗
–	deleted_user_29306	0.6815	✗
–	rakib_hossan	0.6620	✗
–	zulkarnyn420	0.6357	✗
–	loser1	0.5760	✗
–	unknown333	0.5760	✗
–	deleted_user_31920	0.3332	✗
–	rahi_12	0.1210	✗

Table 2: Official ranking of the subtask 1A on the test set. – only participated in the Development Phase. ✗ indicates team has not submitted system in the respective phase. R.: Rank, Dev P.: Development Phase, Eval. P.: Evaluation Phase.

both approaches failed to yield further gains.

Ecstasy (Hasan et al., 2025a) conducted a detailed linguistic analysis of 35,522 Bangla hate speech samples using TF-IDF to identify distinctive lexical patterns for each hate category. Category-specific keywords were embedded into model prompts to provide contextual cues. The model was fine-tuned using LoRA adapters ($r = 64$, $\alpha = 128$) on Llama-3.1-8B with optimized hyperparameters for efficiency. Incorporating keyword-based prompts notably enhanced the model's ability to capture culturally nuanced hate speech patterns unique to Bangla. This team ranked 5^{th} and and 4^{th} in subtask 1A and 1C, respectively.

Gradient Masters (Rahman et al., 2025b) began with BiLSTM and LSTM with attention using pre-trained Bangla embeddings; however, the performance of RNN models prompted a shift to transformer-based models. Their main pipeline fine-tuned BanglaBERT, MuRIL, XLM-R, and DistilBERT with custom classification heads, with BanglaBERT performing best due to language-specific pretraining. To handle severe class imbalance, the authors applied stratified k-fold cross-validation, text normalization, and adversarial training (FGSM). Ensembles of best performing models per subtask were used for final predictions, without post-processing. This team ranked 6^{th} and 3^{rd} in subtask 1A and 1C, respectively.

Catalyst (Hasan and Hasan, 2025) fine-tuned pretrained models such as XLM-RoBERTa (Conneau et al., 2020), mDeBERTa-v3, MuRIL (Khanuja et al., 2021), and IndicBERTv2 (Doddapaneni et al., 2022), optimized using AdamW, mixed-precision training, and task-specific hyperparameters. For single-task setups (e.g., subtask 1A, subtask 1B), authors combined multiple models through hard-voting ensembles to enhance robustness and generalization. For the subtask 1C (multi-task), this system implemented a shared transformer encoder with three task-specific classification heads to jointly predict hate type, severity, and target. Across all subtasks, authors found that multilingual pre-trained transformers and ensembling provided consistent improvements in model stability and performance. This team ranked 7^{th}, 8^{th}, and 10^{th} in subtask 1A, subtask 1B, and subtask 1C, respectively.

BElite (Tripty et al., 2025b) fine-tuned BanglaBERT, mBERT, and XLM-RoBERTa on the dataset and then created an ensemble of these models. Two ensemble strategies were applied: simple averaging and a weighted ensemble, where the weights of individual models were determined based on their weighted F1 scores on the validation

Team	Model														Misc.	
	Classical Model	Neural Networks	BanglaBERT	XLM-RoBERTa	MuRIL	FGSM	IndicBERT	DistilBERT	mE5	mDeBERTa	mBERT	LLMs	Few Shot	Ensemble	Data Preprocessing	Data Augmentation
Code_Gen (Islam et al., 2025)			✓	✓	✓			✓	✓					✓	✓	✓
SyntaxMind (Riad, 2025)		✓	✓									✓		✓	✓	
Ecstasy (Hasan et al., 2025a)		✓	✓	✓	✓			✓								
Gradient Masters (Rahman et al., 2025b)			✓	✓	✓	✓								✓	✓	
Catalyst (Hasan and Hasan, 2025)				✓	✓		✓			✓				✓	✓	
BElite (Tripty et al., 2025b)			✓	✓							✓			✓	✓	
Retriv (Saha et al., 2025)		✓	✓		✓		✓							✓		
CoU-CU-DSG (Alam et al., 2025)			✓	✓	✓											
CUET-NLP_Zenith (Hossan et al., 2025)			✓	✓										✓		
NSU_MILAB (Rahman et al., 2025a)			✓	✓	✓		✓							✓		
PentaML (Tahmid et al., 2025)		✓	✓	✓			✓									
HateNetBN (Anam and Mazumder, 2025)			✓	✓												
Computational StoryLab (Prama et al., 2025)			✓	✓			✓				✓	✓				
Heisenberg (Yasir, 2025)			✓					✓							✓	✓
Bahash-AI (Laskar and Paul, 2025)			✓												✓	✓
Velora (Sayem and Rahman, 2025)			✓												✓	
PerceptionLab (Fahim and Khan, 2025)			✓													✓
PromptGuard (Hossan and Roy Dipta, 2025)												✓	✓		✓	✓
CUET_Sntx_Srfrs (Tripty et al., 2025a)	✓														✓	✓

Table 3: Overview of the approaches used in the submitted systems across three subtasks.

set. The results show that the weighted ensemble outperformed all individual models as well as the simple averaging approach. This team ranked 8^th, 9^{th}, and 5^{th} in subtask 1A, 1B, and 1C, respectively.

Retriv (Saha et al., 2025) employed soft-voting ensembles of transformer models, such as MuRIL, BanglaBERT, and IndicBERTv2, for subtasks 1A and 1B to enhance predictive stability, and a MuRIL-based multi-task framework for subtask 1C to jointly optimize related objectives with inputs truncated to 128 tokens and tuned hyperparameters ($lr = 2e^{-5}$, batch size 16, 3 epochs) applied uniformly. Authors further experimented with hybrid transformer–RNN architectures (BiLSTM, BiGRU) as classification heads to capture sequential context. This team ranked 9^{th}, 10^{th}, and 7^{th} in subtask 1A, 1B, and 1C, respectively.

CoU-CU-DSG (Alam et al., 2025) utilized a weighted probabilistic fusion framework that leverages multiple transformer-based language models for the detection of Bangla hate speech. This approach integrates BanglaBERT, XLM-RoBERTa, and MuRIL, combining their output probabilities through a weighted fusion strategy to leverage the complementary strengths of Bangla-specific and multilingual models. The output of BanglaBERT outperforms other models. This team ranked 10^{th} and 15^{th} in subtask 1A and 1B, respectively.

CUET-NLP_Zenith (Hossan et al., 2025) empliyed a multi-task architecture for Bangla hate speech detection, leveraging a shared transformer backbone with an ensemble of pre-trained models, such as BanglaBERT[10], XLM-RoBERTa, and BanglaBERT (Bhattacharjee et al., 2021). The system jointly classifies hate type, severity, and target group using shared contextual embeddings from the transformer encoder, where text is tokenized to 128 tokens per sequence and processed into 768-dimensional embeddings, followed by [CLS] pooling and dropout regularization. Moreover, task-specific learning rates, a linear scheduler, and summed cross-entropy loss were utilized to fine-tune the model. This team ranked 11^{th}, 13^{th} in subtask 1A and 1B, respectively.

NSU_MILab (Rahman et al., 2025a) evaluated four transformer models, such as BanglaBERT, XLM-RoBERTa, IndicBERT, and Bengali Abusive

[10] https://huggingface.co/sagorsarker/bangla-bert-base

MuRIL, for Bangla hate speech detection. To improve robustness, we applied an ensemble strategy that averaged output probabilities across models, yielding consistent gains over individual systems. Post-competition refinements further confirmed the effectiveness of our ensemble approach in improving overall performance. This team ranked 12^{th} and 17^{th} in subtask 1A and 1B, respectively.

PentaML (Tahmid et al., 2025) fine-tuned multiple pre-trained BERT-based transformer models to classify hate speech in Bangla the comments. To improve performance, PentaML team applied linear probing on three fine-tuned models, allowing better use of learned representations. This lightweight approach achieved consistently better results across all subtasks. This team ranked 14^{th}, 11^{th}, and 13^{th} in subtask 1A, 1B, and 1C, respectively.

HateNetBN (Anam and Mazumder, 2025) utilized parameter-efficient architecture that leverages hierarchical representations from pre-trained transformer models by freezing the backbone to reduce computational cost. A layer-wise attention mechanism learns the relative importance of transformer layers, generating and aggregating context vectors for classification. This design enables effective integration of syntactic and semantic features, providing a lightweight yet powerful alternative to full fine-tuning for Bangla hate speech detection. This team ranked 15^{th} and 12^{th} in subtask 1A and 1B, respectively.

Computational StoryLab (Prama et al., 2025) utilized a multi-task framework built on transformer-based models like BanglaBERT, mBERT, and XLM-RoBERTa. The system uses four separate BERT-based models: a binary classifier to detect toxic comments, and three multiclass classifiers to predict hate type, severity, and target group. Each model includes a dropout layer and a linear output layer, with input processed as standard BERT inputs. Training employed categorical cross-entropy and BCEWithLogitsLoss, optimized with AdamW, while monitoring training and validation accuracy. This team ranked 16^{th} in both subtask 1A and 1B and 11^{th} in subtask 1C.

Heisenberg (Yasir, 2025) tokenized training data using the Bangla basic tokenizer, with stopwords removed. Dataset augmentation included 4,000 newly collected YouTube comments, synonym-based replacement on 8,000 samples, and back-translation of 27,000 samples. This system fine-tuned transformer-based models, including DistillBERT, BanglaHateBERT, BanglaT5, and BanglaBERT. This team ranked 18^{th} in subtask 1A.

Bahash-AI (Laskar and Paul, 2025) used BanglaBERT for all subtasks, applying minimal preprocessing. Subtasks 1A and 1B involved single-label classification, while subtask 1C used a multi-output setup with one-hot encoding and a combined loss to optimize all three labels simultaneously. To increase training size, Bangla texts were translated to English, paraphrased with $pegasus\_paraphrase$, and back-translated, adding 28,220 instances. Models were trained with batch size 16 and dropout 0.1, for 10 epochs with early stopping, and evaluated using F1-score. This team ranked 20^{th} and 17^{th} in subtask 1A and 1B, respectively.

Velora (Sayem and Rahman, 2025) fine-tuned BanglaBERT on a merged dataset combining the competition data with a publicly available Bangla hate speech corpus. To address class imbalance, this system applied back-translation augmentation, logit-adjusted loss, and CB-Focal loss, along with Bangla-specific preprocessing such as NFKC normalization and URL/punctuation removal. Training used a learning rate of 2e-5 (base) and 2e-4 (head), batch size 16, 12 epochs, and early stopping. This team ranked 21^{st} in subtask 1A.

PerceptionLab (Fahim and Khan, 2025) combined Domain-Adaptive Pretraining (DAPT) and multilingual transformers with supervised fine-tuning for hate speech classification. This system augmented the dataset with external corpora and curated examples to address class imbalance. Single-shot six-way classification outperformed hierarchical setups, and DAPT consistently improved performance, especially for majority classes. This team ranked 23^{rd} and 18^{th} in subtask 1A and 1B, respectively.

PromptGuard (Hossan and Roy Dipta, 2025) utilized a few-shot learning approach for Bangla hate speech detection, coordinated by a manager agent. For each input sentence, it generates a few-shot prompt enriched with examples from all six hate categories and category-specific keywords identified via chi-square correlation with the labels. Classification occurs over multiple “turns”, with each turn sampling a new set of examples to create a fresh prompt for inference. The final label is determined by majority voting across turns, with additional iterations used to break ties. This team ranked 30^{th} in the leaderboard of subtask 1A.

CUET_Sntx_Srfrs (Tripty et al., 2025a) evaluated classical machine learning models (LR, DT, MNB, SVC, RF, KNN) using simple and hierarchical pipelines with preprocessing, n-gram features (TF-IDF and Count), and ensemble voting. Hierarchical classification combined with TF-IDF and majority-voting ensembles improved minority class detection while maintaining strong overall performance. This system also assessed the impact of preprocessing and n-gram choices, providing reproducible baselines for Bangla hate speech detection. This team ranked 21^{st} and 18^{th} in subtask 1B and 1C, respectively.

5 Related Work

Detecting offensive language and hate speech has become increasingly crucial with the rapid growth of social media, where harmful content spreads at scale (Jiang and Zubiaga, 2024; Sharma et al., 2022; Alam et al., 2022). Over the past decade, the field has seen a rapid methodological shift (Fortuna and Nunes, 2018), moving from lexicon-based techniques (Waseem and Hovy, 2016) and classical machine learning models (e.g., logistic regression, SVM, etc.) (Davidson et al., 2017) to transformer models and more recently to large language models (LLMs) (Albladi et al., 2025; Hasan et al., 2024).

Early approaches were primarily lexicon-based or relied on shallow statistical models, such as n-gram features with linear classifiers. Subsequent work advanced to deep learning architectures, including recurrent neural networks (e.g., LSTM), and more recently to transformer-based models such as BERT, XLM-R, MuRIL, and AraBERT. Transformer-based models consistently outperform traditional classifiers in the detection of offensive and hate speech (Sharif et al., 2021). Prior work has explored multi-task learning in Arabic (Djandji et al., 2020), code-mixed texts in Dravidian languages (B and A, 2021), and cross-lingual transfer with mBERT and LASER (Pelicon et al., 2021), although cultural biases remain a key limitation (Saumya et al., 2021a).

Kiela et al. (2020) evaluated hateful content detection using SVM, CNN, and LSTM. Multi-label hate speech detection has employed classical models and transformation-based methods (Ibrohim and Budi, 2019), while Mridha et al. (2021) proposed L-Boost, combining BERT embeddings with LSTM for Bangla and Banglish social media. Comparisons on Bangla YouTube and Facebook comments show that SVM often outperforms LSTM and Bi-LSTM (Romim et al., 2021). Hybrid BERT-GRU models have also been applied (Keya et al., 2023), and recent work emphasizes the detection of explainable hate speech (Yang et al., 2023; Piot and Parapar, 2025; Sariyanto et al., 2025).

Several datasets have been developed for hate and offensive content detection. The study of Gupta et al. (2022) created a 150K-comment dataset for Indic languages, and Sharif et al. (2021) studied multilingual code-mixed text, providing baselines for Dravidian languages (Saumya et al., 2021b; Chakravarthi et al., 2022). For Bangla, resources include labeled tweets and comments ranging from 3K to 50K examples (Das et al., 2022a; Romim et al., 2021; Sazzed, 2021; Romim et al., 2022; Das et al., 2022b), while Haider et al. (2024) introduced a multi-label transliterated dataset using LLM-based translation prompting.

Though research on hate speech detection has grown rapidly, deploying these systems in real-world applications remains challenging due to performance gap, cross-lingual transfer, and cultural biases. This shared task aims to advance research through community collaboration and a standardized evaluation framework. As an initial focus, we classified Bangla text samples according to three annotation tasks: Type of Hate, Severity of Hate, and Target of Hate. This framework provides a foundation for future studies, including multi-task learning and explainable hate speech detection.

6 Conclusion

We presented an overview of the Hate Speech Detection shared task at BLP 2025, which focused on identifying hate speech in Bangla social media text across multiple subtasks. Participating systems leveraged transformer-based models, with BanglaBERT, XLM-RoBERTa, and MuRIL being the most widely used, often combined in ensemble setups. Several teams also explored innovative strategies such as few-shot learning, adversarial training, and task-specific loss functions to improve classification. In general, the submissions demonstrated a mix of classical machine learning, neural networks, pretrained language models, and LLMs approaches. For future work, we aim to expand the task to multi-label and multi-modal hate speech detection, as well as develop more robust techniques to handle class imbalance and cultural nuances in Bangla text.

Limitation

The BLP-2025 hate speech detection shared task primarily targeted comment-level classification, which often overlooks broader contextual information. As a result, the identification of nuanced aspects such as hate type, severity, and target remains limited. Additionally, this edition focused exclusively on unimodal (text-only) models, leaving the exploration of multimodal approaches for future research.

Ethics and Broader Impact

The *BanglaMultiHate* dataset contains only textual comments and excludes any personally identifiable information, ensuring that it does not pose direct privacy concerns. However, because annotation is a subjective process, it may still introduce certain biases. To mitigate this risk, a well-defined annotation framework was employed along with comprehensive guidelines to promote consistency and reliability. However, we advise researchers and practitioners to remain aware of these inherent limitations when employing the dataset for modeling or further research.

References

Abdullah Al Maruf, Ahmad Jainul Abidin, Md Mahmudul Haque, Zakaria Masud Jiyad, Aditi Golder, Raaid Alubady, and Zeyar Aung. 2024. Hate speech detection in the bengali language: a comprehensive survey. *Journal of Big Data*, 11(1):97.

Ashraful Alam, Abdul Aziz, and Abu Nowshed Chy. 2025. Cou-cu-dsg at BLP-2025 Task 1: Leveraging weighted probabilistic fusion of language models for bangla hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Firoj Alam, Stefano Cresci, Tanmoy Chakraborty, Fabrizio Silvestri, Dimiter Dimitrov, Giovanni Da San Martino, Shaden Shaar, Hamed Firooz, and Preslav Nakov. 2022. A survey on multimodal disinformation detection. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6625–6643, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Aish Albladi, Minarul Islam, Amit Das, Maryam Bigonah, Zheng Zhang, Fatemeh Jamshidi, Mostafa Rahgouy, Nilanjana Raychawdhary, Daniela Marghitu, and Cheryl Seals. 2025. Hate speech detection using large language models: A comprehensive review. *IEEE Access*.

Mohaymen Ul Anam and Akm Moshiur Rahman Mazumder. 2025. Hatenet-bn at BLP-2025 Task 1: A hierarchical attention approach for bangla hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Bharath B and S. Ajith A. 2021. Ssncse_nlp@dravidianlangtech-eacl2021: Offensive language identification on multilingual code mixing text. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, DravidianLangTech*, pages 313–318. Association for Computational Linguistics.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P McCrae. 2022. Dravidiancodemix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text. *Language Resources and Evaluation*, 56(3):765–806.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

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.

Manjira Das, Sourabh Banerjee, Pushpak Saha, and Animesh Mukherjee. 2022a. Hate speech and offensive language detection in bengali. In *Proceedings of the 4th International Conference on Computational Linguistics (ACLing 2022)*. ArXiv preprint arXiv:2210.03479.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022b. Hate speech and offensive language detection in Bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Susmita Das, Arpita Dutta, Kingshuk Roy, Abir Mondal, and Arnab Mukhopadhyay. 2024. A survey on automatic online hate speech detection in low-resource languages. *arXiv preprint arXiv:2411.19017*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11 of *AAAI '17*.

Mouadh Djandji, Freddy Baly, Wissam Antoun, and Hady Hajj. 2020. Multi-task learning using arabert for offensive language detection. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (OSACT4)*, pages 97–101. European Language Resource Association.

Sumanth Doddapaneni, Rahul Aralikatte, 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 preprint arXiv:2212.05409*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Tamjid Hasan Fahim and Kaif Ahmed Khan. 2025. Perceptionlab at BLP-2025 Task 1: Domain-adapted bert for bangla hate speech detection: Contrasting single-shot and hierarchical multiclass classification. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *Acm Computing Surveys (Csur)*, 51(4):1–30.

Vikram Gupta, Sumegh Roychowdhury, Mithun Das, Somnath Banerjee, Punyajoy Saha, Binny Mathew, Hastagiri Prakash Vanchinathan, and Animesh Mukherjee. 2022. Macd: Multilingual abusive comment detection at scale for indic languages. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, and Md Farhad Alam. 2024. Banth: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Md Sakib Ul Rahman Sourove, Deeparghya Dutta Barua, Md Fahim, and Md Farhad Alam Bhuiyan. 2025. BanTH: A multi-label hate speech detection dataset for transliterated Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7217–7236, Albuquerque, New Mexico. Association for Computational Linguistics.

Kazi Reyazul Hasan, Mubasshira Musarrat, and Muhammad Abdullah Adnan. 2025a. Ecstasy at BLP-2025 Task 1: Tf-idf informed prompt engineering with lora fine-tuning for bangla hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md. Arid Hasan, Shudipta Das, Afiyat Anjum, Firoj Alam, Anika Anjum, Avijit Sarker, and Sheak Rashed Haider Noori. 2024. Zero- and few-shot prompting with LLMs: A comparative study with fine-tuned models for Bangla sentiment analysis. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17808–17818, Torino, Italia. ELRA and ICCL.

Mehedi Hasan and Mahbub Islam Mahim. 2025. Teamhatemate at BLP-2025 Task 1: Divide and conquer: A two-stage cascaded framework with k-fold ensembling for multi-label bangla hate speech classification. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Nahid Hasan and Nahid Hasan. 2025. Catalyst at BLP-2025 Task 1: Transformer ensembles and multi-task learning approaches for bangla hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Md. Refaj Hossan, Kawsar Ahmed, and Mohammed Moshiul Hoque. 2025. Cuet-nlp_zenith at BLP-2025 Task 1: A multi-task ensemble approach for detecting hate speech in bengali youtube comments. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Rakib Hossan and Shubhashis Roy Dipta. 2025. Promptguard at BLP-2025 Task 1: A few-shot classification framework using majority voting and keyword similarity for bengali hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Muhammad Okky Ibrohim and Indra Budi. 2019. Multi-label hate speech and abusive language detection in Indonesian Twitter. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 46–57, Florence, Italy. Association for Computational Linguistics.

Shifat Islam, Emon Ghosh, and Abhishek Agarwala. 2025. Code_gen at BLP-2025 Task 1: Enhancing bangla hate speech detection with transformers through token-aware adversarial contrastive training and layer-wise learning rate decay. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Md Saroar Jahan, Mainul Haque, Nabil Arhab, and Mourad Oussalah. 2022. BanglaHateBERT: BERT for abusive language detection in Bengali. In *Proceedings of the Second International Workshop on Resources and Techniques for User Information in Abusive Language Analysis*, pages 8–15, Marseille, France. European Language Resources Association.

Aiqi Jiang and Arkaitz Zubiaga. 2024. Cross-lingual offensive language detection: A systematic review of datasets, transfer approaches and challenges. *arXiv preprint arXiv:2401.09244*.

A. J. Keya, M. M. Kabir, N. J. Shammey, M. F. Mridha, M. R. Islam, and Y. Watanobe. 2023. G-bert: An efficient method for identifying hate speech in bengali texts on social media. *IEEE Access*, 11:79697–79709.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, and 1 others. 2021. Muril: Multilingual representations for indian languages. *arXiv preprint arXiv:2103.10730*.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. 2020. The hateful memes challenge: Detecting hate speech in multimodal memes. In *Advances in Neural Information Processing Systems (NeurIPS) 33*.

Sahinur Rahman Laskar and Bishwaraj Paul. 2025. Bahash-ai at BLP-2025 Task 1: Bangla hate speech detection using data augmentation and pre-trained model. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Md. Firoj Mridha, Md. Abul Hasnat Wadud, Md. Abdul Hamid, Md. Mostafa Monowar, Md. Abdullah-Al-Wadud, and Atif Alamri. 2021. L-boost: Identifying offensive texts from social media post in bengali. *IEEE Access*, 9:164681–164699.

Anze Pelicon, Raghav Shekhar, Matjaz Martinc, Blaž Škrlj, Matthew Purver, and Simon Pollak. 2021. Zero-shot cross-lingual content filtering: Offensive language and hate speech detection. In *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*, pages 30–34. Association for Computational Linguistics.

Paloma Piot and Javier Parapar. 2025. Towards efficient and explainable hate speech detection via model distillation. In *European Conference on Information Retrieval*, pages 376–392. Springer.

Tabia Tanzin Prama, Christopher M. Danforth, and Peter Sheridan Dodds. 2025. Computational storylab at BLP-2025 Task 1: Hatesense: A transformer-based multi-task learning framework for comprehensive hate speech identification. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Md. Mohibur Nabil Rahman, Muhammad Rafsan Kabir, Rakibul Islam, Fuad Rahman, Nabeel Mohammed, and Shafin Rahman. 2025a. Nsu_milab at BLP-2025 Task 1: Decoding bangla hate speech: Fine-grained type and target detection via transformer ensembles. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Naimur Rahman, Md Sakhawat Hossain, and Syed Mohaiminul Hoque. 2025b. Gradient masters at BLP-2025 Task 1: Advancing low-resource nlp for bengali using ensemble-based adversarial training for hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Md. Shihab Uddin Riad. 2025. Syntaxmind at BLP-2025 Task 1: Leveraging attention fusion of cnn and gru for hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Nauros Romim, Mosahed Ahmed, Md Saiful Islam, Arnab Sen Sharma, Hriteshwar Talukder, and Mohammad Ruhul Amin. 2022. BD-SHS: A benchmark dataset for learning to detect online Bangla hate speech in different social contexts. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5153–5162, Marseille, France. European Language Resources Association.

Nauros Romim, Mosahed Ahmed, Hriteshwar Talukder, and Md Saiful Islam. 2021. Hate speech detection in the bengali language: A dataset and its baseline evaluation. In *Proceedings of International Joint Conference on Advances in Computational Intelligence: IJCACI 2020*, pages 457–468. Springer.

Sourav Saha, K M Nafi Asib, and Mohammed Moshiul Hoque. 2025. Retriv at BLP-2025 Task 1:a transformer ensemble and multi-task learning approach for bangla hate speech identification. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Mohammed Samir, Anisa Meem, and Faisal Abir. 2025. Team_nsu_strugglers at BLP-2025 Task 1: Multilevel approach to detect hateful speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Happy Khairunnisa Sariyanto, Diclehan Ulucan, Oguzhan Ulucan, and Marc Ebner. 2025. Towards explainable hate speech detection. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12883–12893.

S. Saumya, A. Kumar, and J. P. Singh. 2021a. Offensive language identification in dravidian code mixed social media text. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, DravidianLangTech*, pages 36–45. Association for Computational Linguistics.

Sunil Saumya, Abhinav Kumar, and Jyoti Prakash Singh. 2021b. Offensive language identification in Dravidian code mixed social media text. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 36–45, Kyiv. Association for Computational Linguistics.

Sad Yeamin Sayem and Sabira Rahman. 2025. Velora at BLP-2025 Task 1: Multi-method evaluation for hate speech classification in bangla text. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Salim Sazzed. 2021. Abusive content detection in transliterated bengali-english social media corpus. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 125–130, Online. Association for Computational Linguistics.

Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2021. Nlp-cuet@dravidianlangtech-eacl2021: Offensive language detection from multilingual code-mixed text using transformers. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, DravidianLangTech*, pages 255–261, Kyiv. Association for Computational Linguistics.

Deepawali Sharma, Tanusree Nath, Vedika Gupta, and Vivek Kumar Singh. 2025. Hate speech detection research in south asian languages: a survey of tasks, datasets and methods. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 24(3):1–44.

Shivam Sharma, Firoj Alam, Md. Shad Akhtar, Dimitar Dimitrov, Giovanni Da San Martino, Hamed Firooz, Alon Halevy, Fabrizio Silvestri, Preslav Nakov, and Tanmoy Chakraborty. 2022. Detecting and understanding harmful memes: A survey. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, IJCAI '22, pages 5597–5606, Vienna, Austria. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Intesar Tahmid, Rafid Ahmed, Md Mahir Jawad, Anam Borhan Uddin, Md Fahim, and Md Farhad Alam Bhuiyan. 2025. Pentaml @blp shared task 1: Linear probing of pre-trained transformer based models for bangla hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Hafsa Hoque Tripty, Laiba Tabassum, and Hasan Mesbaul Ali Taher. 2025a. Cuet_sntx_srfrs at BLP-2025 Task 1: Combining hierarchical classification and ensemble learning for bengali hate speech detection. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Zannatul Fardaush Tripty, Ibnul Mohammad Adib, Md. Tanjib Hossain, Nafiz Fahad, and Md. Kishor Morol. 2025b. Belite at BLP-2025 Task 1: Leveraging ensemble for multi-task hate speech detection in bangla. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Joseph B Walther. 2022. Social media and online hate. *Current Opinion in Psychology*, 45:101298.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yongjin Yang, Joonkee Kim, Yujin Kim, Namgyu Ho, James Thorne, and Se-Young Yun. 2023. HARE: Explainable hate speech detection with step-by-step reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5490–5505.

Samin Yasir. 2025. Heisenberg at BLP-2025 Task 1: Hate language detection from bangla comments on social media. In *Proceedings of the 2nd Workshop on Bangla Language Processing (BLP 2025)*, Mumbai, India. Association for Computational Linguistics.

Anwar Hossain Zahid, Monoshi Kumar Roy, and Swarna Das. 2025. Evaluation of hate speech detection using large language models and geographical contextualization. *arXiv preprint arXiv:2502.19612*.

A Leaderboard

We report the official leaderboard results for subtask 1B and 1C in Tables 4 and 5.

R.	Team Name	Dev P.	Eval. P.
1	TeamHateMate (Hasan and Mahim, 2025)	0.7536	0.7356
2	Code_Gen (Islam et al., 2025)	0.7532	0.7335
3	Gradient Masters (Rahman et al., 2025b)	0.7496	0.7328
4	Ecstasy (Hasan et al., 2025a)	0.7611	0.7317
5	SyntaxMind (Riad, 2025)	✗	0.7317
6	zannatul_007	0.7508	0.7315
7	abid_al_hossain	0.7448	0.7286
8	Catalyst (Hasan and Hasan, 2025)	0.7456	0.7279
9	BElite (Tripty et al., 2025b)	0.7560	0.7275
10	Retriv (Saha et al., 2025)	0.7500	0.7269
11	PentaML (Tahmid et al., 2025)	✗	0.7256
12	HateNetBN (Anam and Mazumder, 2025)	0.7249	0.7254
13	CUET-NLP_Zenith (Hossan et al., 2025)	0.7333	0.7213
14	adriti12	✗	0.7125
15	CoU-CU-DSG (Alam et al., 2025)	✗	0.7114
16	Computational StoryLab (Prama et al., 2025)	0.7492	0.7095
17	NSU_MILab (Rahman et al., 2025a)	0.7504	0.6981
18	PerceptionLab (Fahim and Khan, 2025)	✗	0.6979
19	pritampal98	0.7337	0.6974
20	Bahash-AI (Laskar and Paul, 2025)	✗	0.6954
21	CUET_Sntx_Srfrs (Tripty et al., 2025a)	✗	0.6817
22	Team_NSU_Strugglers (Samir et al., 2025)	0.6803	0.6760
23	Baseline (Majority)	0.6083	0.5974
24	lamiaa	✗	0.2848
25	Baseline (Random)	0.2118	0.2043
–	manik	0.7393	✗
–	no_name	0.7333	✗
–	Tensoryus	0.7277	✗
–	nur_163	0.7257	✗
–	rabeya_akter	0.7074	✗
–	unknown333	0.6361	✗
–	noob73	0.2647	✗
–	nuralflow	0.0565	✗

Table 4: Official ranking of the subtask 1B on the test set. – only participated in the Development Phase. ✗ indicates the team has not submitted the system in the respective phase. R.: Rank, Dev P.: Development Phase, Eval. P.: Evaluation Phase.

R.	Team Name	Dev P.	Eval. P.
1	TeamHateMate (Hasan and Mahim, 2025)	0.7554	0.7392
2	CUET-NLP_Zenith	0.7520	0.7378
3	Code_Gen (Islam et al., 2025)	0.7558	0.7361
4	Ecstasy (Hasan et al., 2025a)	0.7505	0.7332
5	BElite (Tripty et al., 2025b)	0.7561	0.7312
6	Gradient Masters (Rahman et al., 2025b)	0.7452	0.7310
7	Retriv (Saha et al., 2025)	0.7512	0.7262
8	abid_al_hossain	0.7404	0.7250
9	nur_163	0.7459	0.7241
10	Catalyst (Hasan and Hasan, 2025)	0.7459	0.7240
11	Computational StoryLab (Prama et al., 2025)	✗	0.7233
12	zannatul_007	0.7436	0.7181
13	PentaML (Tahmid et al., 2025)	0.7229	0.7159
14	pritampal98	0.7269	0.7153
15	abir_bot69	✗	0.7129
16	Team_NSU_Strugglers (Samir et al., 2025)	✗	0.7129
17	Bahash-AI (Laskar and Paul, 2025)	✗	0.6969
18	CUET_Sntx_Srfrs (Tripty et al., 2025a)	✗	0.6842
19	aacontest	✗	0.6730
20	Baseline (Majority)	0.6222	0.6072
21	adriti12	0.4141	0.3898
22	Baseline (Random)	0.2300	0.2304
–	foysal_ahmed	0.6939	✗
–	aacontest	0.6838	✗
–	unknown333	0.5669	✗
–	n00b	0.5464	✗

Table 5: Official ranking of the subtask 1C on the test set. – only participated in the Development Phase. ✗ indicates the team has not submitted the system in the respective phase. R.: Rank, Dev P.: Development Phase, Eval. P.: Evaluation Phase.

Bahash-AI at BLP-2025 Task 1: Bangla Hate Speech Detection using Data Augmentation and Pre-trained Model

Sahinur Rahman Laskar
UPES, Dehradun
India
sahinurlaskar.nits@gmail.com

Bishwaraj Paul
Bahash-AI, Bahash Private Limited
Silchar, India
bishwaraj.paul@bahash.in

Abstract

In recent times, internet users are frequently exposed to Hate Speech on social media platforms that have long-lasting negative impacts on their mental wellbeing and also radicalizes the society into an environment of fear and distrust. Many methods have been developed to detect and stop propagation of Hate Speech. However, there is a limitation of annotated data available for Hate Speech in Bengali language. In this work, we have used a pretrained BanglaBERT model on an extended train dataset synthesized via data augmentation techniques. Our team Bahash-AI has achieved 20th, 20th and 17th position of the 3 subtasks out of total 37, 24 and 21 total number of teams who participated in the subtasks 1A, 1B and 1C respectively for Bangla Multi-task Hatespeech Identification Shared Task at BLP Workshop with F1 scores of 0.7028, 0.6954, 0.6969 respectively.

1 Introduction

Hate Speech is generally defined as public speech that conveys hate or incites harm towards an entity that is an individual or a group because of their race, religion, gender, sexual orientation and other characteristics (Cambridge Dictionary).

Hate speech has far-reaching consequences in society that causes the disintegration of social cohesion, incites harm, mental and physical and causes long-lasting psychological effects on individuals and suppresses the development of a nation altogether (Waldron, 2012). Hate speech also causes suppression of free speech of the affected entities, isolating them from public discourse out of fear. A fundamental part of a functioning society is skepticism and debate. People will fear to express new ideas out of fear of being targeted, and the exchange of ideas will become stagnant (Celuch et al., 2023). Political parties and their affiliated individuals and groups use hate speech to retract attention on critical issues and also to divide the population into voting groups to their advantage, hence the existence of hate speech becomes a crisis to the state of democracy (Putra and Damanik, 2021). While some claim that hate speech is protected under free speech and expression, hate speech in turn actually suppresses the expression and participation of free speech of the targeted individuals and communities. Hence, hate speech should be detected to protect societal interests and handled appropriately according to the situation (Waldron, 2012). Due to the advent of social media, it has become hard to detect hate speech manually. Rather, programmatic detection of hate speech has become a critical part of Natural Language Programming (NLP) techniques (Nascimento et al., 2023).

While significant progress has been made in English language due to the huge amount of data available online, Bengali language hate speech is lower as many individuals prefer to use English in social media due to unfamiliarity of using regional language keyboards or even using latinized version of Bengali. There is also the challenge of code-mixing, where English and Bengali are mixed (Das et al., 2022). The main theme of the shared task (Hasan et al., 2025b) is to detect and understand Hate Speech across a variety of subtasks, which is a reflection of real life scenarios where Hate Speech identification requires understanding not just its presence, but also its type, target, and severity.

2 Related Work

We briefly describe the recent related works related to Hate Speech detection techniques. Many works have used classical Machine Learning methods like Logistic Regression, Naive Bayes, Support Vector Machine, Decision Tree, Random Forests, etc. to detect Hate Speech after preprocessing the data with techniques like Lemmatization, Feature Extraction etc. (Davidson et al., 2017). The deep learning models like CNNs, RNNs and transform-

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 402–407
December 23, 2025 ©2025 Association for Computational Linguistics

ers are becoming more prominent nowadays (Malik et al., 2024), (Schmidt and Wiegand, 2017). BanglaBERT is a BERT model that was specifically pretrained on Bengali text to be able to perform tasks for Bengali language inputs (Bhattacharjee et al., 2021). There even exists a BanglaHateBERT specifically designed to detect Hate Speech in Bengali language (Jahan et al., 2022). LLMs are the latest type of models in recent years, which are the current state-of-the-art models. Due to resource unavailability, pretraining full precision LLMs like GPT-OSS, Deepseek R1, Qwen 3, Llama 3.1 etc., which have enormous requirements for training data and GPU resources, becomes unviable at the current stage, but we can use LoRa adapters to finetune a portion of the weights of the model to finetune on the training data albeit with low accuracy score (Albladi et al., 2025). They suggested that future works in LLM specifically on regional languages like Bengali language will possibly pave the way for hate speech detection on full precision LLM models. In our case, due to resource limitations, we have opted for BanglaBERT along with BanglaHateBERT for this task.

3 Dataset

The dataset (Hasan et al., 2025a) [1] used has been provided by the organizers of second Bangla Language Processing (BLP) Workshop for the shared task on Bangla Hate Speech Detection (Hasan et al., 2025b). The dataset consists of 3 subtasks. For all subtasks, each row has a Bengali text collected from YouTube comments. These are then labelled for each subtask appropriately. For subtask 1A, the label is the hate speech category of types *Abusive*, *Sexism*, *Religious Hate*, *Political Hate*, *Profane*, or *None*. For subtask 1B, the label is of the entity towards which the hate speech is directed and is of the following types: *Individuals*, *Organizations*, *Communities*, or *Society*. For subtask 1C, both of the labels from subtask 1A and 1C are included along with a new label categorizing the hate severity and is of the types: *Little to None*, *Mild*, or *Severe*.

The train, validation, devtest and final test consists of 35,522, 2,512, 2,512 and 10,200 instances respectively. For data augmentation, the Bengali train text sets were first translated to English. Next, the pegasus_paraphrase[2] model was used to paraphrase the sentences and then translated back to Bengali again. This added 28,220 instances to the train set.

4 System Description

The BanglaBERT (Bhattacharjee et al., 2022) model was used for the given Hate Speech Detection subtasks with specific finetuning for each of the subtasks. For subtask 1A and 1B, there only had a single output label, so straightforwardly Hugging Face trainer was used to predict the class. Whilst in subtask 1C, there were three outputs to predict, hence the code was modified by outputting the one hot encoded form of all the three labels in a single row, from where the required output was simply extracted. This is simple to implement as it can be simply extracted by slicing, considering the number of unique class labels for each output column. After this, the loss for each output column was calculated and added. By minimizing this combined loss function, the model learns to predict all the output columns correctly simultaneously. Data augmentation has been used to increase train set size, as mentioned in section 3. The model used batch size 16, dropout 0.1 while rest of parameters were set to default values. The models were trained on a single Nvidia L4 GPU with 24GB of VRAM for 10 epochs with early stopping. The training process took about one and half hours for each subtask. F1 score metric was then used to evaluate the models on the basis of the resulting outputs in every subtask. After BanglaBERT models were finetuned, BanglaHateBERT models were finetuned as well, but they had worse performance than BanglaBERT and therefore not submitted for the Shared Task.

5 Result and Analysis

The BLP 2025 shared task organizer (Hasan et al., 2025b) published the evaluation results of the three subtasks of Task 1. The shared task 1 includes three subtasks, namely, subtask 1A: Single label categorization of hate speech type, subtask 1B: Single label categorization of targeted entity of Hate Speech, subtask 1C: Multilabel categorization of hate speech type, severity and targeted entity. Herein, the participation of the research paper was present in all the 3 subtasks 1A, 1B and 1C with a team named Bahash-AI and achieved the 20th, 20th and 17th position of the 3 subtasks out of total 37, 24 and 21 total number of teams who participated

[1] https://github.com/AridHasan/blp25_task1

[2] https://hf.co/tuner007/pegasus_paraphrase

in the subtasks 1A, 1B and 1C respectively.

The evaluation metric used was the F1 score, which was calculated locally using the predictions and the corresponding gold label files provided by the organizers for each subtask. The computation was performed with a simple Python script available in the dataset's GitHub repository, as mentioned in Section 3. For the test set's gold labels which were not provided to us by the organizer before the competition ended, we had to upload the prediction files to Codabench website from where we got our F1 scores. We also compared with our models that were finetuned without data augmentation. The results of our system are marked in Table 1.

Task	Model	Dev	Test
1A	Fine-tuned	0.721	0.693
	Augmentation+ Fine-tuned	**0.724**	**0.703**
1B	Fine-tuned	0.715	0.689
	Augmentation+ Fine-tuned	**0.700**	**0.695**
1C	Fine-tuned	0.701	0.699
	Augmentation+ Fine-tuned	**0.695**	**0.697**

Table 1: BanglaBERT models comparison. The models and figures marked in bold are the latest submitted models for the task.

The n-gram baselines for the subtasks were provided by the organizers as 0.6075, 0.6279 and 0.6401 respectively for the dev-test set. These were thus clearly surpassed by our models, seen from the table 1. Minor improvement was observed on average in models finetuned on augmented data rather than models finetuned on original train dataset.

Now the original and augmented dataset's class label frequency for each subtask have been given in 4, 5 and 2. Since subtask 1C will also have the labels for 1A and 1B, only this class wise frequency is shown, while the rest are in appendix. The class wise accuracy (6) and confusion matrices (1, 2, 3, 4, 5) of test datasets are given in appendix A as well. From these, we can see that due to the None or Little to None categories being a majority proportion of the dataset have the most accuracy. While other categories which are in lower proportions have low accuracy. Sexism for instance in Task 1A and 1C have the lowest counts and hence have the lowest accuracy in both the models with 1A having 0 accuracy and 1C having 0.1034 accuracy. This unbalance ultimately causes the overall accuracy around the range of 0.70 (0.7028, 0.6954, 0.6969). Hence, these models can be said to be very poor in classifying the majority of labels caused by the unbalanced distribution. This could be potentially remedied by having balanced dataset distribution or by increasing data size. As we can see from 1, data augmentation using our method provided limited improvement in accuracy despite nearly doubling the train dataset. Hence, there is need to try other data augmentation procedures or look for further sources of Hate Speech data to include in our training.

Subtask	Label	Original	Augmented
Hate Type	None	19954	35877
	Abusive	8212	14701
	Political Hate	4227	7551
	Profane	2331	4175
	Religious Hate	676	1208
	Sexism	122	221
Hate Severity	Little to None	23489	42199
	Mild	6853	12262
	Severe	5180	9272
To Whom	None	21190	38085
	Individual	5646	10124
	Organization	3846	6909
	Community	2635	4719
	Society	2205	3896

Table 2: Class label frequency for Subtask 1C augmented training data.

Being the case that the augmentation technique used was highly susceptible to translation noise and semantic drift, further experiments were conducted on the original dataset along with oversampling and undersampling methods to get balanced output class frequency. The BanglaHateBERT model mentioned in Section 2 was utilized and finetuned, which is specifically made for Bangla language hate speech. This does solve the issue of highly unbalanced class dataset, but it doesn't yield performance improvement, and hence these models were not submitted to the Shared Task. The results are given in Table 3.

The reason for the BanglaHateBERT model being worse than BanglaBERT on the original dataset need to be investigated further, however it can at least be understood that simple oversampling and undersampling strategies won't be helpful for BanglaBERT model and other synthetic data generation methods have to be explored in future works.

Task	Sampling Strategy	Dev	Test
1A	Original	0.654	0.659
	Oversampled	0.631	0.623
	Undersampled	0.433	0.436
1B	Original	0.664	0.649
	Oversampled	0.531	0.542
	Undersampled	0.612	0.597
1C	Original	0.622	0.607
	Oversampled	0.187	0.192
	Undersampled	0.315	0.311

Table 3: Comparison of different sampling strategies (Original, Oversampling, and Undersampling) for fine-tuning with BanglaHateBERT.

6 Conclusion

This paper showcases our research work in subtask 1, Bangla Hate Speech Detection at BLP Workshop. To tackle the problem of low number of train rows, we have used a data augmentation strategy alongside a specific language oriented pre-trained model, BanglaBERT that shows remarkable accuracy despite the lack of data and training resources. BanglaHateBERT along with oversampling and undersampling methods didn't help in performance improvement compared to original dataset and were worse than BanglaBERT based models. Future work can tackle the problem by introducing more sources of data along with code-mixed and latinized data, more methods of synthetic data generation and more sophisticated models which aligns more with real life situations, which would be specially useful in countries with diverse languages such as India.

References

Aish Albladi, Minarul Islam, Amit Das, Maryam Bigonah, Zheng Zhang, Fatemeh Jamshidi, Mostafa Rahgouy, Nilanjana Raychawdhary, Daniela Marghitu, and Cheryl Seals. 2025. Hate speech detection using large language models: A comprehensive review. *IEEE Access*, 13:20871–20892.

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, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

Cambridge Dictionary. Hate speech. Accessed on 23 September 2025.

Magdalena Celuch, Reetta Oksa, Noora Ellonen, and Atte Oksanen. 2023. Self-censorship among online harassment targets: the role of support at work, harassment characteristics, and the target's public visibility. *Information, Communication & Society*, 27:1–20.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in Bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 512–515.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Md Saroar Jahan, Mainul Haque, Nabil Arhab, and Mourad Oussalah. 2022. Banglahatebert: Bert for abusive language detection in bengali. In *Proceedings of the second international workshop on resources and techniques for user information in abusive language analysis*, pages 8–15.

Jitendra Singh Malik, Hezhe Qiao, Guansong Pang, and Anton van den Hengel. 2024. Deep learning for hate speech detection: a comparative study. *International Journal of Data Science and Analytics*, pages 1–16.

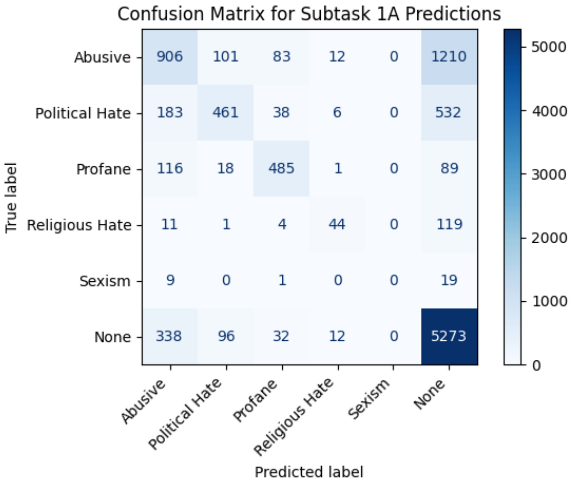


Figure 1: Subtask 1A

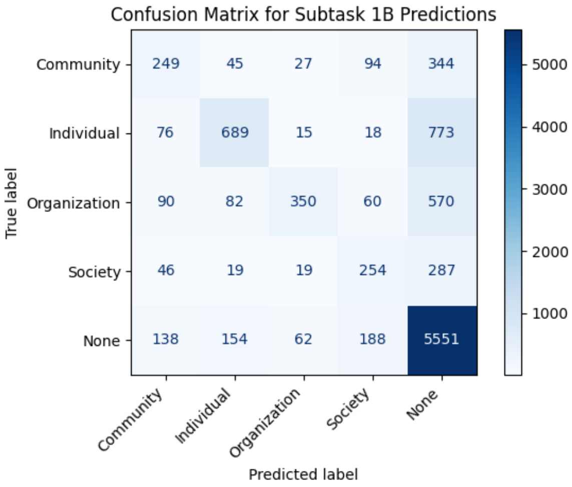


Figure 2: Subtask 1B

Francimaria Nascimento, George Cavalcanti, and Márjory Da Costa-Abreu. 2023. Exploring automatic hate speech detection on social media: A focus on content-based analysis. *SAGE Open*, 13.

Surya Putra and Sisila Damanik. 2021. Hate speech about politics in social media. *Talenta Conference Series: Local Wisdom, Social, and Arts (LWSA)*, 4.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Jeremy Waldron. 2012. *Notes*, pages 235–278. Harvard University Press, Cambridge, MA and London, England.

A Appendix

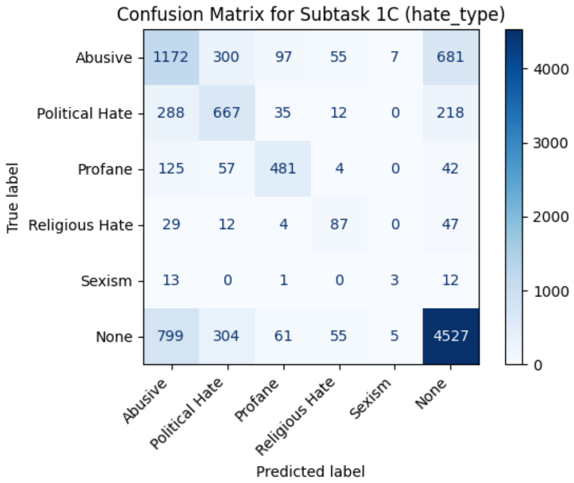


Figure 3: Subtask 1C Hate Type

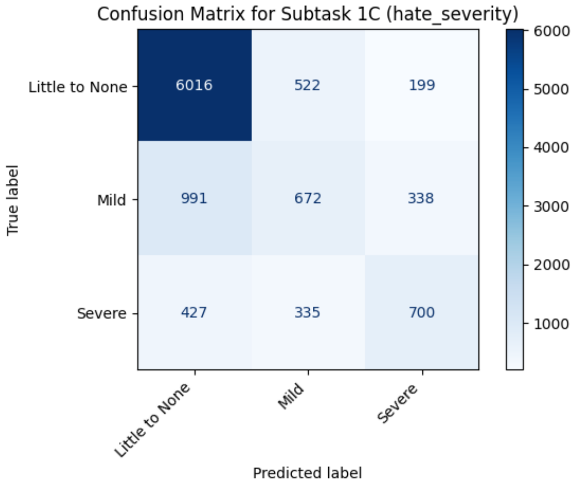


Figure 4: Subtask 1C Hate Severity

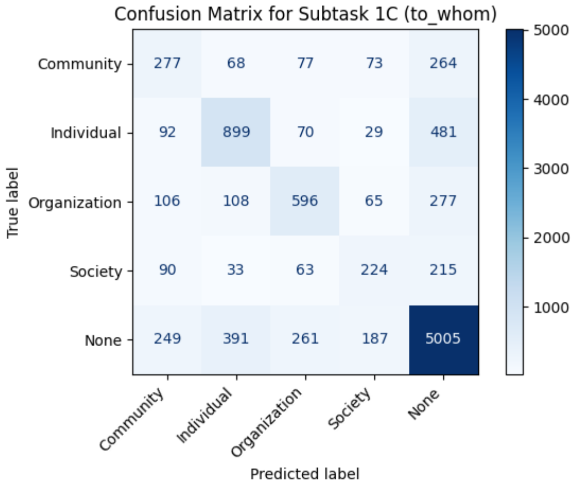


Figure 5: Subtask 1C To Whom

Label	**Original**	**Augmented**
None	19954	35877
Abusive	8212	14701
Political Hate	4227	7551
Profane	2331	4175
Religious Hate	676	1208
Sexism	122	221

Table 4: Subtask 1A: Frequency count of labels before and after augmentation.

Label	**Original**	**Augmented**
None	21190	38085
Individual	5646	10124
Organization	3846	6909
Community	2635	4719
Society	2205	3896

Table 5: Subtask 1B: Frequency count of labels before and after augmentation.

Subtask	**Label**	**Accuracy**
1A	Abusive	0.392
	Political Hate	0.378
	Profane	0.684
	Religious Hate	0.246
	Sexism	0.000
	None	0.917
1B	Community	0.328
	Individual	0.439
	Organization	0.304
	Society	0.406
	None	0.911
1C	**Hate Type**	
	Abusive	0.507
	Political Hate	0.547
	Profane	0.678
	Religious Hate	0.486
	Sexism	0.103
	None	0.787
	Hate Severity	
	Mild	0.336
	Severe	0.479
	Little to None	0.893
	To Whom	
	Community	0.365
	Individual	0.572
	Organization	0.517
	Society	0.358
	None	0.821

Table 6: Label-wise accuracy for different subtasks.

Gradient Masters at BLP-2025 Task 1: Advancing Low-Resource NLP for Bengali using Ensemble-Based Adversarial Training for Hate Speech Detection

Syed Mohaiminul Hoque* and Naimur Rahman* and Md Sakhawat Hossain*

syedmuhaimintahsin@gmail.com
naimur79@student.sust.edu
sakhawatdhrubo@gmail.com

Abstract

This paper introduces the approach of "Gradient Masters" for BLP-2025 Task 1: "Bangla Multitask Hate Speech Identification Shared Task". We present an ensemble-based fine-tuning strategy for addressing subtasks 1A (hate-type classification) and 1B (target group classification) in YouTube comments. We propose a hybrid approach on a Bangla Language Model, which outperformed the baseline models and secured the 6th position in subtask 1A with a micro F1 score of 73.23% and the third position in subtask 1B with 73.28%. We conducted extensive experiments that evaluated the robustness of the model throughout the development and evaluation phases, including comparisons with other Language Model variants, to measure generalization in low-resource Bangla hate speech scenarios and data set coverage. In addition, we provide a detailed analysis of our findings, exploring misclassification patterns in the detection of hate speech.[1]

1 Introduction

Both positive discourse and harmful language are amplified by social media where automatic detection of hate speech is crucial. For low-resource languages such as Bangla, practical moderation is made harder by limited labeled data, extreme class imbalance, and noisy user text which includes mixed scripts, transliteration, and frequent typos. The BLP-2025 Task 1 provides a challenging, fine-grained annotation scheme over YouTube comments comprising both (a) hate-type categories (e.g., Abusive, Political Hate, Sexism) and (b) target groups (e.g., Individual, Organization, Community).

In this work we try to present an ensemble-based fine-tuning pipeline designed to improve robustness and generalization on this task. Our design choices respond to two task-specific challenges:

- **Noise & Orthography:** Bangla social text shows spelling variation and mixed scripts. To tackle this we apply a lightweight rule-based normalization to reduce orthographic variance. before tokenization.

- **Adversarial/noisy input:** User comments often contain typos, slang, or misspellings. In order to improve model resilience, we apply adversarial fine-tuning using FGSM, perturbing token embeddings during training which simulates realistic input variations.

Our contributions are:

1. A reproducible ensemble pipeline for robust Bangla hate speech detection.

2. Empirical analysis of adversarial fine-tuning and normalization effects under severe class imbalance.

3. Error analysis and per-class metrics that identify concrete failure modes for minority categories.

2 Related works

Recent work on hate speech detection in low resource languages has evolved significantly. (Ghosh and Senapati, 2022) analyze transformer-based monolingual and cross-lingual models (e.g., BERT, ALBERT, RoBERTa, DistilBERT, MuRIL) on datasets in Hindi, Marathi, Bangla, Assamese, and new Assamese and Bodo corpora, showing multilingual models generalize better to unseen low-resource languages, though performance varies by language family (Assamese–Bodo vs. Indo-Aryan). (Das et al., 2024) surveys hate speech detection in low-resource languages worldwide, cataloging datasets, features, and methods, highlighting dataset scarcity and linguistic variation as key barriers. In code-mixed settings, (C V et al.,

[1] Our code and models are available at https://github.com/SyedT1/Shared_Task1_HateSpeech

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 408–413
December 23, 2025 ©2025 Association for Computational Linguistics

2024) introduce a multitask framework with XLM-RoBERTa for misinformation and hate speech in Hindi–English posts, demonstrating effectiveness of cross-lingual transformers with tweaks. For Bangla, (Al Maruf et al., 2024) survey linguistic nuances (e.g., dialect, script, informal speech) and challenges in hate speech detection. (Faria et al., 2024) compare LLMs (GPT-3.5 Turbo, Gemini 1.5 Pro) with traditional classifiers in Bangla via zero- and few-shot learning, achieving gains over SVM baselines. New datasets like BIDWESH (Fayaz et al., 2025) and BanTH (Haider et al., 2025) provide multi-dialect or transliterated Bangla corpora with fine-grained labels for hate presence, type, and targets (e.g., gender, religion, origin). Earlier Bangla datasets by (Karim et al., 2020; Romim et al., 2021, 2022) offer 30–50K comments for binary or categorical classification. BLP-2023 shared tasks (Raihan et al., 2023; Tariquzzaman et al., 2023; Fahim, 2023) advance robust systems: (Raihan et al., 2023) use a two-step pipeline for Non-Violence, Passive-Violence, and Direct-Violence with back-translation and multilinguality; (Tariquzzaman et al., 2023) combine Informal Bangla FastText embeddings with BiLSTM for cost-effective performance near transformers; (Fahim, 2023) enhance BanglaBERT via in-task pretraining, adversarial perturbation, and ensembles for sentiment, applicable to multi-task hate classification. Our work builds on these by integrating type, target, and severity in Bangla YouTube comments, emphasizing fine-grained labels and efficient models for low-resource settings.

3 Dataset Description

The dataset for this task (Hasan et al., 2025b,a) exhibits significant class imbalance across both subtasks, with the "None" label constituting the majority in both training and development sets. Table 1 presents the distribution for Subtask 1A (Type of Hate) and Table 2 shows the distribution for Subtask 1B (Target Group).

Distribution analysis reveals consistent patterns between training and development sets, indicating proper data splitting. In particular, severe class imbalance poses challenges for model training, particularly for minority classes in Subtask 1A such as Sexism (0. 34% in training) and Religious Hate (1.90% in training). Similarly, in Subtask 1B, the None class constitutes 59.66% of the training set, with minority classes like Society (6.21%) and Community (7.42%).

Label	Train	Dev	Dev Test	Test
None	19,954	1,451	1,447	5,751
Abusive	8,212	564	549	2,312
Political Hate	4,227	291	283	1,220
Profane	2,331	157	185	709
Religious Hate	676	38	40	179
Sexism	122	11	8	29
Total	35,522	2,512	2,512	10,200

Table 1: Subtask 1A (Type of Hate) Dataset Distribution

Label	Train	Dev	Dev Test	Test
None	21,190	1,536	1,528	6,093
Individual	5,646	364	391	1,571
Organization	3,846	292	292	1,152
Community	2,635	179	159	759
Society	2,205	141	142	625
Total	35,522	2,512	2,512	10,200

Table 2: Subtask 1B (Target Group) Dataset Distribution

4 Method Description

4.1 Adversarial Fine-tuning (FGSM)

We use the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) as a light-weight adversarial fine-tuning mechanism to improve robustness to small input perturbations (typos, obfuscation, or transliteration noise common in YouTube comments). FGSM perturbs token embeddings in the direction of the loss gradient:

$$\Delta = \epsilon \cdot \mathrm{sign}(\nabla_{\Theta} J(x, y; \Theta)) \qquad (1)$$

and we minimize a combined loss:

$$\tilde{J}(x, y) = \alpha J(x, y) + (1 - \alpha) J(x + \Delta, y). \qquad (2)$$

In our experiments we used $\epsilon = 0.1$ and $\alpha = 0.5$, applying FGSM on perturbed embeddings every other epoch during fine-tuning. We report these hyperparameters and ablate FGSM vs. standard fine-tuning in Section 5. FGSM is applied in embedding space only (not at token substitution level), which is computationally inexpensive compared to stronger iterative attacks.

While stronger iterative methods like Projected Gradient Descent (PGD) (Geisler et al., 2024) and Adversarial Weight Perturbations(AWP) (Wu et al., 2020) could yield more robust perturbations, we

[0] **Dev** – public development set (hyper-parameter tuning);
Dev Test – blind validation set (used for leaderboard);
Test – official blind test set (final ranking).
This footnote applies to both Table 1 and Table 2.

selected FGSM for its single-step efficiency, reducing training time by 50% compared to PGD and 57% compared to AWP respectively(based on preliminary trials).

4.2 Data Augmentation

To enhance our model's performance,we applied external augmentation by collecting multiple labeled hate speech samples from public datasets such as WoNBias (Aupi et al., 2025) and Bangla Hate Speech Detection Dataset (Romim et al., 2021) for subtask 1A. These datasets were specifically selected to address categories like sexism, religious hate, and political hate. Initially, training on this combined dataset yielded improved accuracy on the development set. However, the same model performed poorly on the test set, likely due to domain mismatch or overfitting to the augmented data. As a result, we reverted to training our final models solely on the original training dataset.

5 Results and Analysis

We present comprehensive experimental results for both subtasks, demonstrating the effectiveness of our proposed ensemble-based approach with adversarial training and text normalization.

5.1 Subtask 1A: Hate Speech Type Classification

Table 3 summarizes our experimental results for multi-class classification of Bengali text into six categories: Abusive, Sexism, Religious Hate, Political Hate, Profane, or None.

Our analysis shows several key findings: (1) Traditional deep learning models (BiLSTM, LSTM+Attention) significantly underperformed compared to transformer-based approaches, with F1 scores around 55-56%. (2) Among base LLMs, XLM-R-large achieved the highest development performance (72.81%), followed by MuRIL-large (71.02%) and BanglaBERT (70.74%). (3) K-fold cross-validation (KF) improved performance by 2-3% across models, with MuRIL-large+KF reaching 73.61% on Dev and 71.88% on Test. (4) Text normalization (N) further boosted performance, particularly for BanglaBERT+N (74.32% Dev, 71.14% Test). (5) The combination of BanglaBERT with K-fold, FGSM adversarial training, and normalization (BanglaBERT+KF+FGSM+N) achieved the highest scores of 74.88% on Dev and 72.33% on Test, showing the effectiveness of combining these techniques.

Model	Dev	Test
DL Models		
BiLSTM	56.25	47.64
LSTM+Attention	55.18	54.41
Base LLMs		
XLM-R-large	72.81	70.54
MuRIL-large	71.02	70.29
BanglaBERT	70.74	70.31
BanglaBERT-large	70.51	68.13
XLM-R-base	70.50	70.15
DistilBERT-multi	68.03	68.49
K-Fold CV		
MuRIL-large+KF	73.61	71.88
XLM-R-large+KF	73.45	71.72
BanglaBERT+KF	73.29	72.05
K-Fold CV + Normalizer		
BanglaBERT+N	74.32	71.14
XLM-R-large+N	73.29	71.57
MuRIL+N	73.73	72.30
Adversarial attacks w/o Normalizer		
BanglaBERT+KF+FGSM	73.87	72.17
MuRIL+KF+FGSM	73.68	71.90
Adversarial Attacks w Normalizer		
BanglaBERT+KF+FGSM+N	**74.88**	**72.33**
MuRIL+KF+FGSM+N	73.81	71.31

Table 3: Subtask 1A Results (Micro F1 %)

5.2 Subtask 1B: Target Group Classification

Table 4 presents results for hate speech target classification into four categories: Individuals, Organizations, Communities, or Society.

For Subtask 1B, we observe: (1) Base transformer models showed consistent performance, with F1 scores ranging from 69.27% to 72.09% on Dev and 68.46% to 71.23% on Test. (2) MuRIL-large with K-fold cross-validation obtained the best scores of 74.96% on Dev and 73.44% on Test, indicating strong generalization. (3) Text normalization improved performance, with BanglaBERT+N and MuRIL+N reaching 74.72% and 74.48% on Dev, respectively, and 72.89% and 73.44% on Test. (4) Adversarial training with FGSM further enhanced results, with XLM-R-large with KFold CV and FGSM attack achieving 74.20% on Dev and 73.28% on Test. (5) The combination of K-fold, FGSM, and normalization (e.g., BanglaBERT+KF+FGSM+N) yielded competitive performance (74.64% Dev, 73.12% Test), though slightly below MuRIL-large+KF on Test.

5.3 Error Analysis

Tables 3 and 4 show that BanglaBERT with FGSM adversarial training and text normalization obtains the best performance for Subtask 1A (72.33% test F1), while MuRIL-large with K-fold achieved the best for Subtask 1B (73.44% test F1). To gain further insight about system performance, the con-

Model	Dev	Test
Base LLMs		
BanglaBERT	72.09	70.25
MuRIL-large	71.93	70.93
XLM-R-large	71.38	71.23
BanglaBERT-large	69.90	68.88
XLM-R-base	71.09	70.19
DistilBERT-multi	69.27	68.46
K-Fold CV		
MuRIL-large+KF	**74.96**	**73.44**
BanglaBERT+KF	73.69	71.85
XLM-R-large+KF	71.53	68.07
K-Fold CV + Normalizer		
BanglaBERT+N	74.72	72.89
MuRIL+N	74.48	73.44
XLM-R-large+N	72.39	71.66
Adversarial attacks w/o Normalizer		
BanglaBERT+KF+FGSM	74.12	72.25
MuRIL+KF+FGSM	73.89	72.92
XLM-R-large+KF+FGSM	74.20	73.28
Adversarial attacks w Normalizer		
BanglaBERT+KF+FGSM+N	74.64	73.12
MuRIL+KF+FGSM+N	74.56	72.95
XLM-R-large+KF+FGSM+N	74.32	72.17

Table 4: Subtask 1B Results (Micro F1 %)

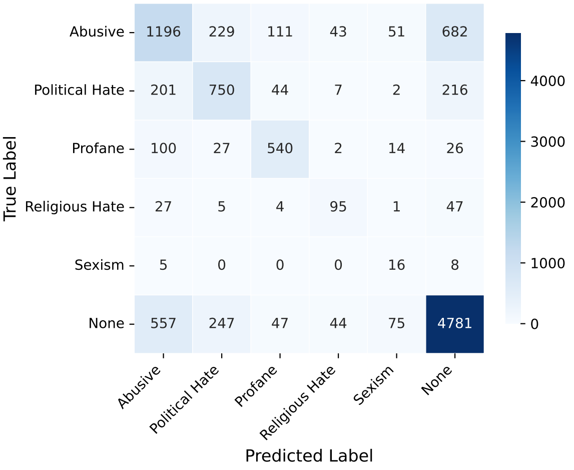


Figure 1: Confusion matrix analysis for Subtask 1A using BanglaBERT with FGSM attack

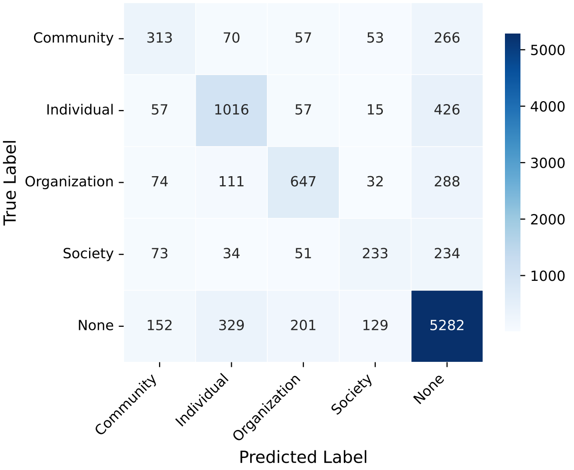


Figure 2: Confusion matrix analysis for Subtask 1B using MuRiL with K Fold CV

fusion matrices (Figures 1 and 2) are analyzed.

For Subtask 1A (Figure 1), the model obtains the highest True Positive Rate (TPR) of 83.13% (4781/5751) for None category and 76.24% (540/709) for Profane category. The model shows moderate performance for Political Hate with 61.48% (750/1220) TPR and Religious Hate with 53.07% (95/179) TPR. However, it provides the lowest TPR of 55.17% (16/29) for Sexism category.

Our model significantly misclassified Abusive texts, where only 51.73% (1196/2312) were correctly identified. A major portion of Abusive texts were misclassified as None (682 instances) and Political Hate (229 instances). This suggests the model struggles to distinguish between general abusive language and other hate categories, likely due to overlapping linguistic features such as aggressive vocabulary and derogatory terms. Political Hate texts also showed considerable confusion, with 216 instances misclassified as Abusive and 201 as None. The class imbalance, with Sexism representing only 0.34% of training data, contributes to poor minority class performance.

For Subtask 1B (Figure 2), the model achieves the highest TPR of 86.72% (5282/6093) for None category and 64.59% (1016/1571) for Individual targeting. The model shows moderate performance for Organization with 56.16% (647/1152) TPR. However, it provides lower TPR for Community (41.24%, 313/759) and Society (37.28%, 233/625) categories.

The confusion between Individual and None categories shows 426 Individual texts misclassified as None, indicating difficulty identifying subtle personal attacks lacking explicit hate markers. Community-targeted hate shows significant confusion with None (266 misclassifications), while Society-targeted hate is frequently misclassified as None (234 instances). Organization texts are often confused with None (288 instances) and Individual (111 instances). This pattern suggests that abstract group-targeted criticism is difficult to distinguish from neutral text or personal attacks. The imbalanced dataset, where None represents 59.66% of training data, biases toward the majority class, causing systematic misclassification of minority categories.

5.4 Computational Analysis

All experiments were conducted on dual NVIDIA T4 GPUs (T4x2 configuration) which were available through Kaggle and Google Colab platforms. Base LLM fine-tuning (e.g., BanglaBERT) required 2-3 hours per epoch. The full pipeline with K-fold cross-validation (5 folds) and FGSM for BanglaBERT took approximately 6-7 hours in total. For larger models like MuRIL-large and XLM-R-large with K-fold and FGSM, training exceeded 12 hours, hitting the session limits on these platforms and requiring session restarts. FGSM added 10-15% overhead per epoch due to gradient computations but remained lightweight compared to iterative attacks like PGD.

6 Conclusion

Our experiments demonstrated the efficacy of integrating multilingual pre-trained models with robustness-enhancing techniques, especially in addressing class imbalances and noisy social media text from YouTube comments. The use of adversarial perturbations made the model more resilient to real-world changes like typos and informal language. Normalization, on the other hand, fixed script inconsistencies that are common in Bangla online content.

Limitations

Our approach has some limitations even though the results are promising. The dataset has severe class imbalance, particularly for minority classes like "Sexism" and "Religious Hate" in subtask 1A, whereas we observed the same for "Community" and "Society" in subtask 1B. This might have led to under-performance on these categories, even with K-fold cross-validation and adversarial training. While we experimented with external data augmentation, it did not yield improvements on the test set, suggesting potential domain mis-matches between public datasets and the YouTube-specific corpus for the shared task.

Additionally, we intended to explore more advanced adversarial techniques, such as Geometry-Aware Adversarial Training (GAT) (Zhu et al., 2022) and Adversarial Weight Perturbation (AWP) (Wu et al., 2020), to further enhance robustness of the models. However, these were infeasible due to computational constraints as fine-tuning larger models like XLM-RoBERTa-large and MuRIL-large uncased required approximately 5 hours per epoch. This exceeded the 12-hour session limits on platforms like Kaggle and Google Colab causing frequent restarts and complicating experimentation.

References

A. Al Maruf, J. J. Abidin, M. M. Haque, Z. M. Jiyad, A. Golder, R. Alubady, and Z. Aung. 2024. Hate speech detection in the bengali language: a comprehensive survey. *Journal of Big Data*, 11(1).

Md. Raisul Islam Aupi, Nishat Tafannum, Md. Shahidur Rahman, Kh Mahmudul Hassan, and Naimur Rahman. 2025. WoNBias: A dataset for classifying bias & prejudice against women in Bengali text. In *Proceedings of the 6th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 105–110, Vienna, Austria. Association for Computational Linguistics.

Sunil Gopal C V, Sudhan S, Shreyas Gutti Srinivas, Sushanth R, and Abhilash C B. 2024. Faux-hate multitask framework for misinformation and hate speech detection in code-mixed languages. In *Proceedings of the 21st International Conference on Natural Language Processing (ICON): Shared Task on Decoding Fake Narratives in Spreading Hateful Stories (Faux-Hate)*.

S Das, A Dutta, K Roy, A Mondal, and A Mukhopadhyay. 2024. A survey on automatic online hate speech detection in low-resource languages. *arXiv preprint*, arXiv:2411.19017.

M. Fahim. 2023. Aambela at BLP-2023 task 2: Enhancing BanglaBERT performance for Bangla sentiment analysis task with in task pretraining and adversarial weight perturbation. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 317–323.

F. T. J. Faria, L. H. Baniata, and S. Kang. 2024. Investigating the predominance of large language models in low-resource bangla language over transformer models for hate speech detection: A comparative analysis. *Mathematics*, 12(23):3687.

A. H. Fayaz, M. S. Uddin, R. U. Bhuiyan, Z. Sultana, M. S. Islam, B. Paul, T. Muhammad, and S. Manzoor. 2025. Bidwesh: A bangla regional based hate speech detection dataset. *arXiv preprint arXiv:2507.16183*.

Simon Geisler, Tom Wollschläger, Mohamed Hesham Ibrahim Abdalla, Johannes Gasteiger, and Stephan Günnemann. 2024. Attacking large language models with projected gradient descent. *arXiv preprint arXiv:2402.09154*.

Koyel Ghosh and Apurbalal Senapati. 2022. Hate speech detection: a comparison of mono and multilingual transformer model with cross-language evaluation. In *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

F. Haider, F. T. Shifat, M. F. Ishmam, D. D. Barua, M. S. U. R. Sourove, M. Fahim, and M. F. Alam. 2025. Banth: A multi-label hate speech detection dataset for transliterated bangla. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Blp 2023 task 1: Hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

M. R. Karim, C. Chakraborty, B. R. Chakravarthi, J. P. McCrae, and M. Cochez. 2020. Classification benchmarks for under-resourced bengali language based on multi-channel convolutional-lstm network. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*.

M. N. Raihan, D. Goswami, S. S. C. Puspo, and M. Zampieri. 2023. nlpbdpatriots at blp-2023 task 1: Two-step classification for violence inciting text detection in bangla – leveraging back-translation and multilinguality. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 179–184.

N. Romim, M. Ahmed, A. S. Sharma, H. Talukder, and M. R. Amin. 2022. Bd-shs: A benchmark dataset for learning to detect online bangla hate speech in different social contexts. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5153–5162.

Nauros Romim, Mosahed Ahmed, Hriteshwar Talukder, and Md Saiful Islam. 2021. Hate speech detection in the bengali language: A dataset and its baseline evaluation. In *Proceedings of International Joint Conference on Advances in Computational Intelligence: IJCACI 2020*, pages 457–468. Springer.

M. Tariquzzaman, M. W. Kader, A. Anam, N. Haque, M. Kabir, H. Mahmud, and M. K. Hasan. 2023. the_linguists at blp-2023 task 1: A novel informal bangla fasttext embedding for violence inciting text detection. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 214–219.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. 2020. Adversarial weight perturbation helps robust generalization. *Advances in neural information processing systems*, 33:2958–2969.

Bin Zhu, Zhaoquan Gu, Le Wang, Jinyin Chen, and Qi Xuan. 2022. Improving robustness of language models from a geometry-aware perspective. *arXiv preprint arXiv:2204.13309*.

$\mathcal{P}$rompt$\mathcal{G}$uard at BLP-2025 Task 1: A Few-Shot Classification Framework Using Majority Voting and Keyword Similarity for Bengali Hate Speech Detection

Rakib Hossan
Bangladesh University of Business
and Technology, Bangladesh
rakib911hossan@gmail.com

Shubhashis Roy Dipta
University of Maryland, Baltimore
County, USA
sroydip1@umbc.edu

Abstract

Content Warning: *This paper contains content that some readers may find inappropriate or disturbing.*

The BLP-2025 Task 1A requires Bengali hate speech classification into six categories. Traditional supervised approaches need extensive labeled datasets that are expensive for low-resource languages. We developed $\mathcal{P}$rompt$\mathcal{G}$uard, a few-shot framework combining chi-square statistical analysis for keyword extraction with adaptive majority voting for decision-making. We explore statistical keyword selection versus random approaches and adaptive voting mechanisms that extend classification based on consensus quality. Chi-square keywords provide consistent improvements across categories, while adaptive voting benefits ambiguous cases requiring extended classification rounds. $\mathcal{P}$rompt$\mathcal{G}$uard achieves 67.61 micro-F1, outperforming n-gram baselines (60.75) and random approaches (14.65). Ablation studies confirm chi-square based keywords show the most consistent impact across all categories. [1]

1 Introduction

The widespread presence of hate speech on online platforms threatens user safety worldwide. While Natural Language Processing has advanced through transformer models (Vaswani et al., 2017), pre-trained models like BERT (Devlin et al., 2019), and large language models (Brown et al., 2020), Bengali faces unique challenges in automated content moderation due to its rich morphology and culturally specific expressions of toxicity. Additionally, the field suffers from a lack of large-scale annotated datasets (Romim et al., 2021; Das and Mukherjee, 2023). The BLP-2025 Task 1A (Hasan et al., 2025a,b) requires classifying Bengali text into six categories: none, sexism, abusive, profane, religious hate, and political hate. This task comes with some significant challenges: severe class imbalance with limited training data for Bengali hate speech detection (Faria et al., 2024), the scarcity of comprehensive datasets (Al Maruf et al., 2024), and resource constraints common to low-resource language processing (Magueresse et al., 2020).

Although Bengali hate speech detection has been explored in prior work, most systems fail to generalize under data imbalance and overlapping linguistic cues between categories such as abusive and profane (Romim et al., 2021; Das and Mukherjee, 2023). We introduce $\mathcal{P}$rompt$\mathcal{G}$uard, a statistically grounded few-shot framework combining: (1) chi-square analysis for discriminative keyword extraction (Azzahra et al., 2021), (2) systematic two-phase example selection inspired by few-shot learning principles (Brown et al., 2020), and (3) adaptive majority voting (Dietterich, 2000) that extends decisions when consensus is unclear. Our approach demonstrates that few-shot methods combining statistical feature selection and adaptive voting can achieve competitive performance without requiring extensive labeled datasets.

To summarize, our contributions are: (1) novel integration of chi-square feature selection with few-shot prompting, (2) adaptive majority voting with dynamic decision extension, and (3) comprehensive ablation studies validating each component's effectiveness.

2 $\mathcal{P}$rompt$\mathcal{G}$uard

$\mathcal{P}$rompt$\mathcal{G}$uard combines chi-square keyword extraction, adaptive example selection, and majority voting with consensus extension to achieve robust Bengali hate speech classification without extensive labeled datasets. $\mathcal{P}$rompt$\mathcal{G}$uard combines chi-square keyword extraction, adaptive example selection, and majority voting with consensus ex-

[1] https://github.com/Rakib911Hossan/PromptGuard

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 414–420
December 23, 2025 ©2025 Association for Computational Linguistics

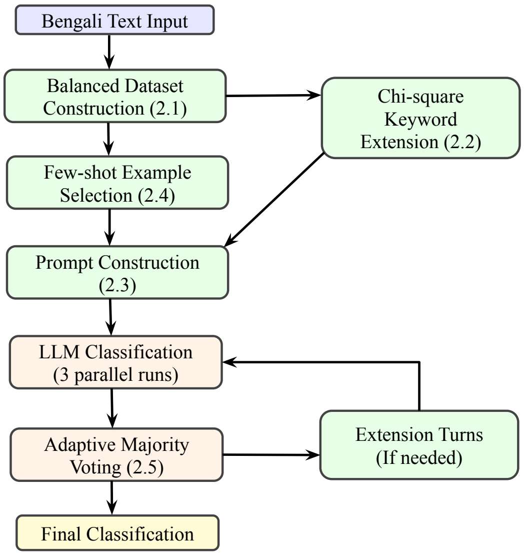

Bengali Text Input
Balanced Dataset Construction (2.1)
Chi-square Keyword Extension (2.2)
Few-shot Example Selection (2.4)
Prompt Construction (2.3)
LLM Classification (3 parallel runs)
Adaptive Majority Voting (2.5)
Extension Turns (If needed)
Final Classification

Figure 1: Overall architecture of PromptGuard framework. The system processes Bengali text through balanced dataset construction, extracts discriminative keywords using chi-square analysis, constructs prompts with selected examples, performs parallel LLM classifications, and applies adaptive majority voting to produce the final hate speech category classification.

tension to achieve robust Bengali hate speech classification without extensive labeled datasets. The framework operates through an iterative decision-making process (illustrated in Fig. 1): it begins by running 3 parallel classification attempts, each using different example sets. If these initial attempts produce a clear majority vote (>50% agreement), the winning classification is returned immediately. However, when no clear consensus emerges, the system adaptively extends the voting process by adding additional classification rounds with randomly selected examples. This extension continues iteratively until either a clear majority is achieved or a maximum of 10 total turns is reached, at which point the system selects from among the tied labels. This adaptive mechanism proves particularly valuable for ambiguous cases where the initial classification attempts disagree. The complete method pipeline is illustrated in Fig. 1.

2.1 Balanced Dataset Construction

The first step creates a balanced training pool by sampling 120 examples per category, resulting in 720 examples across 6 categories. This size was constrained by the smallest category (sexism) to ensure balanced representation without data augmentation. Random sampling with a fixed seed ensures reproducibility and prevents category bias.

Category	**Top-2 Keywords**	χ^2 **Score**
Profane	বাল	2980.27
	মাগির	1559.10
Political Hate	ভোট	1738.07
	বিএনপি	1534.63
Religious Hate	মুসলিম	1378.39
	হিন্দু	1280.53
Sexism	নারী	871.51
	পরকিয়া	801.57

Table 1: Top-2 discriminative keywords per hate speech category ranked by chi-square scores. Profane category shows the highest statistical association (χ^2 = 2980.27), while sexism exhibits the lowest scores among hate categories (χ^2 = 871.51).

2.2 Statistical Keyword Extraction using Chi-Square Testing

We have used the chi-square (χ^2) to identify words most statistically associated with each hate speech category (Forman, 2003), ensuring genuine discriminative power over arbitrary selection. Table 1 presents the highest-scoring keywords for each category based on chi-square analysis. The preprocessing pipeline applies Unicode filtering, minimum document frequency of 5, and maximum of 95% to retain discriminative Bengali vocabulary.

Manual Keyword Refinement The statistically extracted keywords undergo manual filtering for cultural sensitivity while maintaining balanced representation across categories. The refined keyword sets include:

- **Abusive:** দালাল, টিভি, ফালতু, চোর, মিথ্যা, পাগল, জুতা, লজ্জা, আমিন
- **Profane:** বাল, মাগি, খানকি, বেশ্যা, দফা, বাচ্চা, সালা, শালা, মাদারচোদ, কুত্তা, জারজ, পোলা, শুয়োর
- **Religious Hate:** মুসলিম, হিন্দু, ইহুদি, মুসলমান, গজব, ধর্ম, ইসলাম, কাফের, মসজিদ, ধর্মীয়, মোল্লা, আল্লাহ
- **Political Hate:** ভোট, বিএনপি, আওয়ামী, লীগ, সরকার, নির্বাচন, হাসিনা, অবৈধ, জনগণ, পার্টি, দল, চোর, রাজনীতি

Model	Random	Majority	n-gram	**$\mathcal{P}$rompt$\mathcal{G}$uard**	Rank-1	Rank-2	Rank-3
micro-f1	14.65	57.60	60.75	67.61	73.62	73.45	73.40

Table 2: $\mathcal{P}$rompt$\mathcal{G}$uard performance compared to baselines and top-performing systems on 10,000 test instances. While outperforming simple baselines by 7-53 micro-F1 points, a 6-point gap remains with leading approaches.

- **Sexism:** নারী, পরকিয়া, মহিলা, পুরুষ, হিজরা, বিয়ে, লিঙ্গ, হোটেল, মেয়ে, বেডা, আবাসিক

2.3 Enhanced Prompt Engineering

The method uses a structured prompt (§A.1) template that combines statistical keywords with few-shot examples to improve classification accuracy. Specifically, the prompt integrates three key components: (1) Examples for each category, (2) Keywords from each category, (3) step-by-step reasoning guidelines.

2.4 Dynamic Few-Shot Learning Strategy

The example selection method uses two phases to balance diversity, as sample selection strategies significantly impact few-shot learning performance (Pecher et al., 2024). In the first phase (3 turns), examples are selected sequentially from a pool of 120 instances without reuse, using 20 examples per category per turn. For future turns, if needed, the system extends voting for up to 10 additional turns using random selection of examples to provide fresh perspectives on ambiguous cases. If consensus remains unclear after maximum iterations, the system selects one random winner from the tied ones (the label with the highest vote count) as the final decision.

2.5 Robust Majority Voting with Adaptive Extension

The voting mechanism operates in two phases building on self-consistency approaches (Wang et al., 2022):

Initial Phase: Execute $n_0 = 3$ parallel voting turns $\{V_1, V_2, V_3\}$, each using distinct example sets E_i where $E_i \cap E_j = \emptyset$ for $i \neq j$.

Adaptive Extension: If no clear majority emerges, iteratively add voting turns V_{3+k} for $k = 1, 2, \ldots, 10$, where each V_{3+k} uses freshly sampled examples $E_{3+k} \sim \mathcal{D}$. The process terminates when $\max_c |S_c| > \frac{1}{2}|S|$ or maximum iterations are reached, where S_c represents votes for candidate c.

3 Results

3.1 Experimental Setup

We use `Qwen/Qwen3-32B` with temperature=0 and parallel processing for efficiency. Following the official shared task, we have used `micro-f1` as the main evaluation. To give more insights into the result, we have also provided confusion matrix with fine-grained analysis for each metric. All experiments were conducted on a single NVIDIA L40S (46 GB) GPU, and the full evaluation on the test set required approximately four hours.

3.2 Discussion

As a baseline, we have provided the random, majority and n-gram baseline. To compare with other participants' work, we have also reported the 1st, 2nd and 3rd results from the official leaderboard. We have reported the results on the Table 2. While $\mathcal{P}$rompt$\mathcal{G}$uard achieves better scores than the original baselines, still it lags behind compared to the Rank-{1,2,3} models.

To perform a fine-grained analysis of class-wise performance, we present the confusion matrix in Fig. 2. The breakdown reveals that the model exhibits a bias toward labeling instances as non-hate, more than any other category. The poorest performance is observed in the "Abusive" category. One possible reason for this is that many of the keywords associated with the Abusive class, i.e., টিভি, মিথ্যা, পাগল, জুতা, লজ্জা, আমিন – can also appear in benign, non-hateful contexts. This semantic overlap makes the classification of the "Abusive" category particularly challenging.

4 Ablation Studies

Due to the high computation and time needed to run on the whole test set (10k), we sampled a balanced subset from the test set. In the test set, the sexism category has the lowest number of examples (29). We therefore sampled a balanced subset of 174 instances (29 per label).

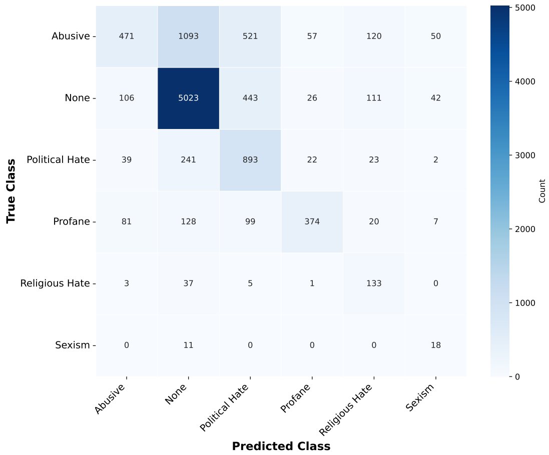


Figure 2: Per-category breakdown reveals uneven performance across hate speech types. Non-hate content is classified most accurately, while abusive language detection proves most challenging.

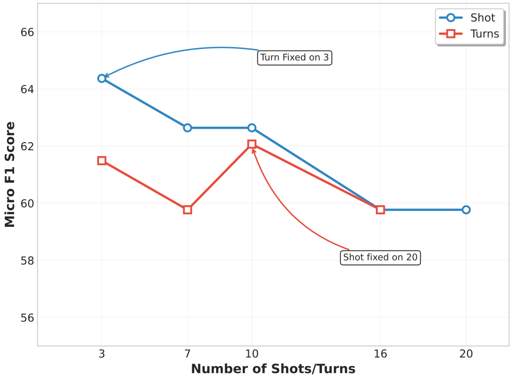


Figure 3: Impact of varying the number of examples and turns on final evaluation performance. We observe a negative correlation with the number of examples, likely attributable to the "lost-in-the-middle" problem (Liu et al., 2023).

4.1 Impact of Similar Keywords

As described in §A, our final model utilizes a well-crafted prompt that includes both in-context examples and targeted keywords. To assess the contribution of these keywords, we compare performance against a baseline using a basic prompt without the keywords. Both prompt versions are provided in §A.

On our sampled test dataset, the keyword-free prompt achieves a micro-F1 score of 57.47, whereas the prompt with keywords reaches 59.77, highlighting the effectiveness and importance of including targeted keywords in the prompt design.

4.2 Impact of Shots & Turns

In our original configuration, we used 20 examples per label in the prompt and 3 initial turns. To evaluate the impact of these two parameters, we conducted a controlled analysis by varying one while keeping the other fixed. Specifically, we experimented with {3, 7, 10, 16, 20} shots and {3, 7, 10, 16} turns. The results of this ablation are presented in Fig. 3.

The results indicate that the best performance is achieved with 3 shots and 3 turns. We hypothesize that this is due to the "lost-in-the-middle" effect in LLMs (Liu et al., 2023), where too many in-context examples can reduce the model's focus on relevant inputs. In contrast, varying the number of turns has minimal impact on performance. This aligns with our expectations, as the initial turns are primarily used to identify a clear winner; additional turns are also invoked if the earlier ones fail to produce a confident outcome.

4.3 Impact of Different Models

In our primary results, we used the `Qwen3-32B` model. To evaluate the impact of model architecture and size, we further ran our method on two families of models: Qwen3 (Yang et al., 2025) and GPT-OSS (OpenAI et al., 2025), across varying model sizes. The comparative results are presented in Fig. 4.

The findings show a clear positive correlation between model size and micro-F1 score within the Qwen3 family, indicating that larger models yield better performance. In contrast, the GPT-OSS models display relatively smaller performance variation across sizes and consistently lag behind the best-performing Qwen models.

5 Related Works

Bangla Hate Speech Detection. Karim et al. (2021) achieved F1-scores of 78-91% using transformer ensembles, while Jahan et al. (2022) developed domain-specific BanglaHateBERT with 1.5M Bengali posts. Raihan et al. (2023) addressed code-mixed content through cross-lingual transformers for transliterated text.

LLM in Bengali. Hasan et al. (2023); Roy Dipta and Vallurupalli (2024) showed that monolingual transformers outperform general-purpose LLMs (e.g., Flan-T5, GPT-4) on zero/few-shot Bangla sentiment tasks, while Wang et al. (2025) improved LLM performance through multilingual prompting enriched with cultural cues. To address reasoning limitations in LLMs, Colelough

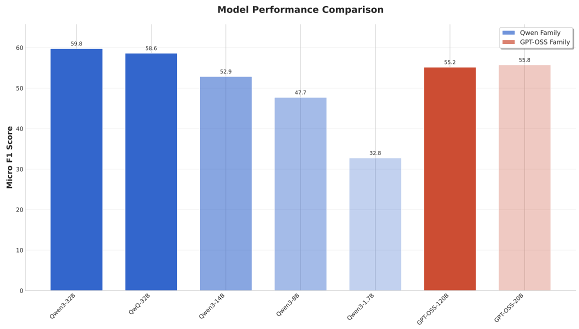


Figure 4: Performance comparison across model architectures and sizes. Qwen3 models show clear scaling benefits with size, while GPT-OSS models exhibit consistent but lower performance across different scales.

and Regli (2025) identified major gaps in explainability and meta-cognition across Neuro-Symbolic AI literature. In parallel, Kowsher et al. (2022) introduced Bangla-BERT with language-specific pretraining, outperforming prior models by up to 5.3%. Gao et al. (2025) tackled prompt design challenges with MAPS, an automated framework achieving higher coverage via diversity-guided search.

Social Media Challenges. Guo et al. (2024) categorized LLM biases and proposed mitigation strategies, while Natsir et al. (2023) examined dynamic language evolution in social media, identifying shifts in multilingual adaptation.

6 Conclusion

We present $\mathcal{P}$rompt$\mathcal{G}$uard, a few-shot classification framework that addresses Bengali hate speech detection through statistical feature selection and adaptive decision-making. Our approach combines chi-square-based keyword extraction with majority voting to achieve robust classification. Our work demonstrates the value of integrating statistical foundations with few-shot learning for low-resource language tasks. Future directions include exploring advanced feature selection methods to improve few-shot hate speech detection.

References

Abdullah Al Maruf, Ahmad Jainul Abidin, Md Mahmudul Haque, Zakaria Masud Jiyad, Aditi Golder, Raaid Alubady, and Zeyar Aung. 2024. Hate speech detection in the bengali language: a comprehensive survey. *Journal of Big Data*.

Nadhia Azzahra, Danang Murdiansyah, and Kemas Lhaksmana. 2021. Toxic comment classification on social media using support vector machine and chi square feature selection. *International Journal on Information and Communication Technology (IJoICT)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*.

Brandon C Colelough and William Regli. 2025. Neurosymbolic ai in 2024: A systematic review. *arXiv preprint arXiv:2501.05435*.

Mithun Das and Animesh Mukherjee. 2023. Banglaabusememe: A dataset for bengali abusive meme classification. *arXiv preprint arXiv:2310.11748*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*.

Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer.

Fatema Tuj Johora Faria, Laith H Baniata, and Sangwoo Kang. 2024. Investigating the predominance of large language models in low-resource bangla language over transformer models for hate speech detection: A comparative analysis. *Mathematics*.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*.

Shuzheng Gao, Chaozheng Wang, Cuiyun Gao, Xiaoqian Jiao, Chun Yong Chong, Shan Gao, and Michael Lyu. 2025. The prompt alchemist: Automated llm-tailored prompt optimization for test case generation. *arXiv preprint arXiv:2501.01329*.

Yufei Guo, Muzhe Guo, Juntao Su, Zhou Yang, Mengqiu Zhu, Hongfei Li, Mengyang Qiu, and Shuo Shuo Liu. 2024. Bias in large language models: Origin, evaluation, and mitigation. *arXiv preprint arXiv:2411.10915*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing*

(BLP-2025), India. Association for Computational Linguistics.

Md Arid Hasan, Shudipta Das, Afiyat Anjum, Firoj Alam, Anika Anjum, Avijit Sarker, and Sheak Rashed Haider Noori. 2023. Zero-and few-shot prompting with llms: A comparative study with fine-tuned models for bangla sentiment analysis. *arXiv preprint arXiv:2308.10783*.

Md Saroar Jahan, Mainul Haque, Nabil Arhab, and Mourad Oussalah. 2022. Banglahatebert: Bert for abusive language detection in bengali. In *Proceedings of the second international workshop on resources and techniques for user information in abusive language analysis*, pages 8–15.

Md Rezaul Karim, Sumon Kanti Dey, Tanhim Islam, Sagor Sarker, Mehadi Hasan Menon, Kabir Hossain, Md Azam Hossain, and Stefan Decker. 2021. Deephateexplainer: Explainable hate speech detection in under-resourced bengali language. In *2021 IEEE 8th international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE.

Md Kowsher, Abdullah As Sami, Nusrat Jahan Prottasha, Mohammad Shamsul Arefin, Pranab Kumar Dhar, and Takeshi Koshiba. 2022. Bangla-bert: transformer-based efficient model for transfer learning and language understanding. *IEEE Access*, 10:91855–91870.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. *arXiv preprint*. ArXiv:2307.03172 [cs].

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

Nurasia Natsir, Nuraziza Aliah, Zulkhaeriyah Zulkhaeriyah, Amiruddin Amiruddin, and Farida Esmianti. 2023. The impact of language changes caused by technology and social media. *Language Literacy: Journal of Linguistics, Literature, and Language Teaching*.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Branislav Pecher, Ivan Srba, Maria Bielikova, and Joaquin Vanschoren. 2024. Automatic combination of sample selection strategies for few-shot learning. *arXiv preprint arXiv:2402.03038*.

Md Nishat Raihan, Umma Hani Tanmoy, Anika Binte Islam, Kai North, Tharindu Ranasinghe, Antonios Anastasopoulos, and Marcos Zampieri. 2023. Offensive language identification in transliterated and code-mixed bangla.

Nauros Romim, Mosahed Ahmed, Hriteshwar Talukder, and Md Saiful Islam. 2021. Hate speech detection in the bengali language: A dataset and its baseline evaluation. In *Proceedings of International Joint Conference on Advances in Computational Intelligence: IJCACI 2020*. Springer.

Shubhashis Roy Dipta and Sai Vallurupalli. 2024. UMBCLU at SemEval-2024 task 1: Semantic textual relatedness with and without machine translation. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1351–1357, Mexico City, Mexico. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*.

Qihan Wang, Shidong Pan, Tal Linzen, and Emily Black. 2025. Multilingual prompting for improving llm generation diversity. *arXiv preprint arXiv:2505.15229*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

A Prompts

A.1 Prompt with Keywords

Following is the prompt that we have used with our final method.

You are an AI language model specialized in detecting hate speech in Bengali. Your task is to classify a given Bengali sentence into one of six categories: none, sexism, abusive, profane, religious hate, or political hate.

First, review these examples of sentences for each category:
<examples>
{{EXAMPLES}}
<examples>

Now, consider these common words associated with each category. Note that the presence of these words doesn't guarantee classification into that category, but they can be helpful indicators:

<category_keywords>
abusive: দালাল, টিভি, ফালতু, চোর, মিথ্যা, পাগল, জুতা, লজ্জা, আমিন
profane: বাল, মাগি, খানকি, বেশ্যা, দফা, বাচ্চা, সালা, শালা, মাদারচোদ, কুত্তা, জারজ, পোলা, শুয়োর
religious hate: মুসলিম, হিন্দু, ইহুদি, মুসলমান, গজব, ধর্ম, ইসলাম, কাফের, মসজিদ, ধর্মীয়, মোল্লা, আল্লাহ
political hate: ভোট, বিএনপি, আওয়ামী, লীগ, সরকার, নির্বাচন, হাসিনা, অবৈধ, জনগণ, পার্টি, দল, চোর, রাজনীতি
sexism: নারী, পরকিয়া, মহিলা, পুরুষ, হিজরা, বিয়ে, লিঙ্গ, হোটেল, মেয়ে, বেডা, আবাসিক
<category_keywords>

Here's the Bengali sentence you need to classify:
<input_sentence>
{{INPUT_SENTENCE}}
<input_sentence>

Before making your final classification, analyze the sentence in detail. Consider the following:
1. Compare the sentence to the provided examples.
2. Examine the tone, specific words used, and overall context.
3. Check if any words from the category_keywords are present and relevant.
4. For each category (none, sexism, abusive, profane, religious hate, political hate):
    - List evidence for classifying the sentence into this category.
    - List evidence against classifying the sentence into this category.
5. Summarize your findings and explain your final decision.

Your final output should be the category classification. Use only one of these exact category names: none, sexism, abusive, profane, religious hate, or political hate.

Provide your classification inside <classification> tags.

A.2 Basic Prompt

Following is the prompt that we used during the ablation study of similar keywords.

You are an AI language model specialized in detecting hate speech in Bengali. Your task is to classify a given Bengali sentence into one of six categories: none, sexism, abusive, profane, religious hate, or political hate.

First, review these examples of sentences for each category:
<examples>
{{EXAMPLES}}
<examples>

Here's the Bengali sentence you need to classify:
<input_sentence>
{{INPUT_SENTENCE}}
</input_sentence>

Before making your final classification, analyze the sentence in detail. Consider the following:
1. Compare the sentence to the provided examples.
2. Examine the tone, specific words used, and overall context.
3. For each category (none, sexism, abusive, profane, religious hate, political hate):
    - List evidence for classifying the sentence into this category.
    - List evidence against classifying the sentence into this category.
4. Summarize your findings and explain your final decision.

Your final output should be the category classification. Use only one of these exact category names: none, sexism, abusive, profane, religious hate, or political hate.

Provide your classification inside <classification> tags.

BElite at BLP-2025 Task 1: Leveraging Ensemble for Multi Task Hate Speech Detection in Bangla

Zannatul Fardaush Tripty[1,*], **Ibnul Mohammad Adib**[2,*], **Nafiz Fahad**[3]
Muhammad Tanjib Hussain[3] **Md Kishor Morol**[3,†]
[1] Chittagong University of Engineering and Technology
[2]American International University of Bangladesh, [3]Elite Research Lab LLC
Correspondence: kishormorol@ieee.org

Abstract

The widespread use of the internet has made sharing information on social media more convenient. At the same time, it provides a platform for individuals with malicious intent to easily spread hateful content. Since many users prefer to communicate in their native language, detecting hate speech in Bengali poses a significant challenge. This study aims to identify Bengali hate speech on social media platforms. A shared task on Bengali hate speech detection is organized by the Second Bangla Language Processing Workshop (BLP). To tackle this task, we implement five traditional machine learning models (LR, SVM, RF, NB, XGB), three deep learning models (CNN, BiLSTM, CNN+BiLSTM), and three transformer-based models (BanglaBERT, m-BERT, XLM-R). Among all models, a weighted ensemble of transformer models achieves the best performance. Our approach ranks ***3rd*** in Subtask 1A with a micro-*F1* score of 0.734, ***6th*** in Subtask 1B with 0.7315, and, after post-competition experiments, ***4th*** in Subtask 1C with 0.735.

1 Introduction

The rise of social media has enabled billions to share opinions but has also fueled online hate speech, defined as speech inciting hatred against groups based on ethnicity, religion, disability, or sexual identity (American Library Association, 2017). With that comes the caveat of manual content moderation, requiring the development of state-of-the-art content moderation by leveraging artificial intelligence and natural language processing (Amin, 2024).

Most research has focused on high-resource languages like English, while Bangla, despite being the sixth most spoken language globally, remains under-resourced for NLP tasks (Hosain and Morol, 2025; Salam et al., 2016). It poses unique linguistic sociocultural challenges, especially for hate speech detection, including code-switching, spelling variation, and dialect variation. To address this, a shared task on Bangla hate speech classification is organized at BLP (Hasan et al., 2025b), providing a labeled dataset and dividing the task into three subtasks: hate type, severity, and target group, reflecting the complexity of understanding and mitigating hate speech (Hossan et al., 2025; Islam et al., 2024). In this work, we develop an ensemble model for Bangla hate speech classification and conduct thorough experiments across all three subtasks. Our approach demonstrates improved accuracy compared to baseline methods, addressing the gap in NLP research in Bangla.

2 Related Works

Early works in hate speech detection introduced the first annotated dataset, where a GRU with word2vec embeddings outperformed several machine learning models, demonstrating deep learning's superiority (Ishmam and Sharmin, 2019; Hosain et al., 2025a). Subsequent studies moved toward context-aware neural architectures. A two-part encoder–decoder framework with 1D CNNs, BiRNNs, and attention layers was proposed (Das et al., 2021), showing that attention mechanisms outperformed standalone traditional deep learning (Zerine et al., 2020).

Later on, the landscape expanded with BD-SHS, a large benchmark dataset for binary and multi-label classification, which introduced informal fastText embeddings tailored for noisy social media text, highlighting the role of domain-specific representation learning (Romim et al., 2022).

Furthermore, domain-specific embeddings were shown to capture hateful vocabulary better than general-purpose embeddings; therefore, even lighter models were able to rival transformers—an

[*]Equal contribution.
[†]Corresponding author.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 421–429
December 23, 2025 ©2025 Association for Computational Linguistics

important insight for resource-constrained environments (Saleh et al., 2023; Tariquzzaman et al., 2023).

The rise of transformers advanced Bangla hate detection, with monolingual BanglaBERT often outperforming multilingual encoders such as XLM-R and mBERT (Ghosh and Senapati, 2022). Transformer models were further tested with Romanized Bangla compared to standard Bangla, showing that MuRIL excelled in cross-lingual few-shot settings (Das et al., 2022).

Domain-specific transformer models such as BanglaHateBERT (Jahan et al., 2022) yielded consistent gains, while hybrid architectures such as G-BERT (Keya et al., 2023) combined BanglaBERT with a GRU classifier, further improving performance. DeepHateExplainer (Karim et al., 2021) was another pioneering effort with an ensemble of various BERT-based models (Hosain et al., 2025b). Furthermore, it used layer-wise propagation and sensitivity analysis to provide explanations and ensure that the model's decisions were made based on reasonable features; however, it also highlighted the need for more contextual information, especially for labels such as political hate speech.

Beyond Bangla, multi-task learning with user metadata and inter-user or intra-user features improved English hate detection, suggesting that a similar approach could benefit Bangla (Kapil and Ekbal, 2024). Recent explorations with large and small language models (LLMs) also signaled a paradigm shift. GPT-3.5 Turbo with chain-of-thought prompting was shown to outperform BERT baselines on English hate speech; however, performance varied across different languages (Guo et al., 2023; Sakib et al., 2025). TinyLLMs (Phi-2, TinyLlama) fine-tuned with LoRA were also shown to rival large language models in efficiency and accuracy (Sen et al., 2024).

Hate speech detection has remained challenging because datasets vary in annotation quality, domain, and class distribution. Furthermore, creating reliable resources is costly, as it requires strong agreement among annotators to reduce bias (Vasker et al., 2024; Gupta et al., 2025).

3 Task and Dataset Description

The primary objective of this task is to detect hate speech in a Bengali corpus by developing systems capable of accurately classifying text, using the datasets (Hasan et al., 2025a) provided by the organizers of the shared task.[1] A significant class imbalance is observed across all three subtasks of the BLP 2025 dataset, as reflected in Tables 1, 2, and 3. For Subtask 1A (Hate Type Classification), shown in Table 1, the *None* class overwhelmingly dominates the dataset, while several hate categories contain very few examples. In particular, *Sexism* and *Religious Hate* account for only a small portion of the training and evaluation splits, making them the most underrepresented classes. For Subtask 1B (Target Classification: To Whom), as presented in Table 2, the *None* category remains the most frequent, similar to Subtask 1A. In contrast, the minority classes—especially *Community* and *Society*—have far fewer samples. For Subtask 1C, which focuses on multi-task classification, an additional column representing *hate severity* is included. The goal of this subtask is to perform multi-task classification, assigning each text a hate type, a target (to whom), and a severity level. The distribution of hate severity across the dataset is shown in Table 3.The majority of instances fall under the *Little to None* severity level, while the *Severe* category appears sparsely across all splits.

Classes	**Train**	**Dev**	**Test**
None	19954	1451	5751
Abusive	8212	564	2312
Political Hate	4227	291	1220
Profane	2331	157	709
Religious Hate	676	38	179
Sexism	122	11	29
Total	35522	2512	10200

Table 1: Dataset distribution across classes, splits, and word counts for Subtask 1A (hate type)

Class	**Train**	**Dev**	**Test**
None	21190	1536	6093
Individual	5646	364	1571
Organization	3846	292	1152
Community	2635	179	759
Society	2205	141	625
Total	35522	2512	10200

Table 2: Dataset distribution across classes, splits, and word counts for Subtask 1B (to whom)

[1] https://github.com/AridHasan/blp25_task1

Hate Severity	Train	Dev	Test
Little to None	23489	1703	6737
Mild	6853	483	2001
Severe	5180	326	1462
Total	35522	2512	10200

Table 3: Dataset distribution across hate severity classes for train, dev, and test splits.

4 Methods

To evaluate the performance of Bengali hate speech classification, we implements five machine learning methods (LR, RF, NB, SVM, and XGB), three deep learning techniques (CNN, BiLSTM, and CNN+BiLSTM), and three transformer-based models (mBERT, Bangla-BERT, XLM-R, along with an ensemble strategy). An abstract overview of the system is illustrated in Figure 1.

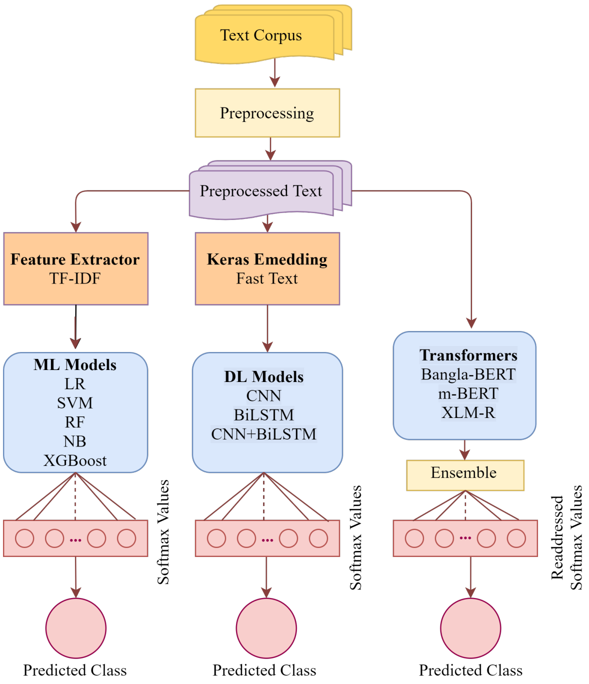


Figure 1: Abstract process diagram of Bengali hate speech detection system.

4.1 Preprocessing

As the dataset is relatively clean, only minimal preprocessing was applied. Text normalization is performed using the Bangla Normalizer tool (Hasan et al., 2020), which standardizes spacing, punctuation, and character representations for consistency.

4.2 Machine Learning Models

All five ML models use unigram features extracted via TF-IDF. For instance, Logistic Regression (LR) employs the `liblinear` solver with $l2$ regularization ($C = 5.0$), 500 iterations, and balanced class weights. Meanwhile, Random Forest (RF) uses 300 trees with the `gini` criterion, considers all features, and requires at least two samples to split a node. In contrast, Naive Bayes (NB) applies additive smoothing ($\alpha = 0.5$) to optimize probability estimates. Similarly, the Support Vector Machine (SVM) uses a linear kernel with $C = 2$, balanced class weights, and 500 iterations to ensure convergence. Finally, XGBoost (XGB) is configured for multi-class classification with 500 trees and a learning rate of 0.1. These settings are carefully chosen to maximize accuracy and stability across all classifiers.

4.3 Deep Learning Models

We implement three deep learning models: CNN, BiLSTM, and hybrid CNN+BiLSTM. All models use pretrained FastText embeddings (Joulin et al., 2016) to obtain dense word representations. The CNN comprises two convolutional blocks, with 128 filters of size 5 and 64 filters of size 3, each followed by max-pooling layers of sizes 5 and 3, respectively. The features are then flattened and passed through a dense layer with 128 *ReLU*-activated neurons, followed by a dropout layer with rate 0.5 and a softmax output layer. The BiLSTM uses a bidirectional LSTM with 200 units and dropout 0.2, followed by a dense layer with 128 *ReLU*-activated neurons, dropout 0.5, and a softmax output. The hybrid CNN+BiLSTM first applies the CNN convolutional and pooling layers, then feeds the resulting features into a bidirectional LSTM with 200 units and dropout 0.2. The output is flattened, passed through a dense layer with 128 *ReLU* neurons, a dropout layer with rate 0.5, and a softmax layer for classification.

4.4 Transformer Models

Past studies show that transformer models trained in monolingual, multilingual, or cross-lingual settings achieve state-of-the-art performance in hate speech classification (Mazari et al., 2024; Saleh et al., 2023). In this work, we select Bangla-BERT, XLM-R, and mBERT for our ensemble because they represent complementary architectures that have demonstrated strong performance in Bangla

NLP tasks. Bangla-BERT (Bhattacharjee et al., 2022) is a monolingual model that effectively captures language-specific morphology and culturally grounded expressions, making it particularly suitable for Bangla hate speech detection. XLM-R (Conneau et al., 2020) is a cross-lingual model that provides robust representations across multiple languages and handles noisy or code-mixed social media text effectively. mBERT (Devlin et al., 2019), although trained on a smaller multilingual corpus, has shown strong generalization across low-resource languages, including Bangla. All three models are pre-trained transformers that we fine-tune on the shared-task dataset, accessed via the Hugging Face library[2] .

The models are fine-tuned on the dataset using the Hugging Face Trainer API[3] for 3 epochs, with a batch size of 8 for both training and evaluation. A learning rate of $2e^{-5}$ and a weight decay of 0.01 are applied. Evaluation and model checkpointing are performed every 500 steps, and the best-performing model on the validation set is automatically loaded at the end of training.

4.5 Proposed Ensemble Model

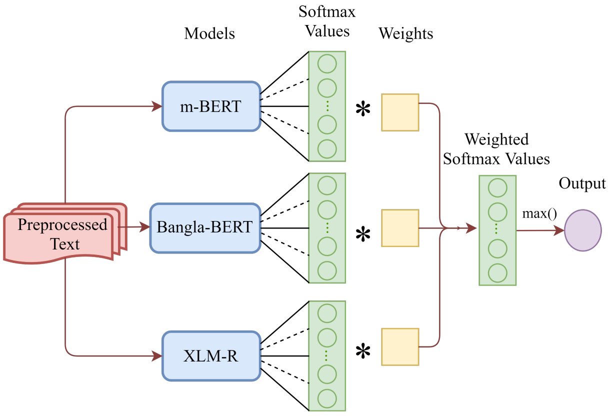


Figure 2: Proposed ensemble method.

Recent studies (Karim et al., 2021; Singh et al., 2023) have demonstrated that ensembles of transformer models can substantially improve the effectiveness of classification tasks. Ensemble learning leverages the complementary strengths of individual models to enhance overall predictive accuracy. In this work, three pretrained transformer models—Bangla-BERT, XLM-R, and m-BERT—are fine-tuned on their respective datasets and subsequently combined using both averaging *(A-ensemble)* and weighted *(W-ensemble)* approaches. The average ensemble computes the mean of the softmax probabilities generated by the participating models and assigns the class with the highest probability as the final prediction. In contrast, weighted ensemble combines the predictions of multiple models by assigning different importance (weights) to each model based on their prior performance. In this experiment, the weighted ensemble assigns weights based on micro-*F1* scores from the evaluation dataset. These weights are normalized and combined with the softmax probabilities generated by the fine-tuned BERT models, thereby allowing models with superior prior performance to exert greater influence on the final prediction. The overall process of the weighted ensemble is illustrated in Figure 2.

Let $M = \{M_1, M_2, \ldots, M_L\}$ represent the set of fine-tuned models, where $L = 3$ in our case. For a given instance, let $p_i(c)$ denote the softmax probability predicted by model M_i for class c, and let f_i be the micro-*F1* score of model M_i on the evaluation dataset.

We first compute normalized weights for each model based on their micro-*F1* scores:

$$w_i = \frac{f_i}{\sum_{j=1}^{L} f_j}, \quad i = 1, 2, \ldots, L \tag{1}$$

The weighted ensemble probability for class c is then computed as:

$$P(c) = \sum_{i=1}^{L} w_i \cdot p_i(c) \tag{2}$$

Finally, the predicted class $\hat{y}$ is determined as the class with the highest weighted probability:

$$\hat{y} = \arg\max_{c} P(c) \tag{3}$$

This approach ensures that models with higher prior performance (as measured by micro-*F1*) contribute more to the final prediction, while still leveraging the complementary strengths of all models in the ensemble.

5 Experiments and Results

The evaluation results of individual models on the test set are presented in Table 4. The results indicate that among the machine learning approaches, XGB with TF-IDF features achieved the highest micro *F1*-scores, recording 0.66, 0.67, and 0.68 for

[2] https://huggingface.co/

[3] https://huggingface.co/docs/transformers/main_classes/trainer

	Subtask 1A			Subtask 1B			Subtask 1C		
Models	P	R	F1	P	R	F1	P	R	F1
TF-IDF+LR	0.65	0.66	0.65	0.65	0.65	0.65	0.66	0.67	0.67
TF-IDF+SVM	0.63	0.64	0.64	0.64	0.63	0.63	0.65	0.65	0.65
TF-IDF+RF	0.64	0.65	0.65	0.63	0.66	0.66	0.64	0.67	0.67
TF-IDF+NB	0.60	0.61	0.61	0.57	0.63	0.63	0.60	0.64	0.64
TF-IDF+XGB	0.65	0.67	0.66	0.64	0.67	0.67	0.65	0.68	0.68
FT+CNN	0.65	0.68	0.68	0.63	0.67	0.67	0.65	0.69	0.69
FT+BiLSTM	0.67	0.69	0.69	0.66	0.69	0.69	0.69	0.70	0.70
FT+C+B	0.66	0.68	0.68	0.64	0.68	0.68	0.66	0.69	0.69
m-BERT	0.68	0.70	0.70	0.69	0.70	0.70	0.67	0.69	0.69
XLM-R	0.70	0.69	0.69	0.70	0.71	0.71	0.70	0.71	0.71
Bangla-BERT	0.72	0.72	0.72	0.72	0.72	0.72	0.72	**0.74**	0.73
W-Ensemble	**0.72**	**0.73**	**0.734**	**0.72**	**0.73**	**0.731**	**0.72**	0.72	**0.735**
A-Ensemble	0.71	0.71	0.71	0.70	0.71	0.71	0.70	0.71	0.72

Table 4: Performance of various models on the Subtask 1A and Subtask 1B test sets where P, R, and F denote precision, recall, and micro F1-score, respectively. Here, C+B represents the CNN+BiLSTM model and FT represents FastText.

Subtasks 1A, 1B, and 1C, respectively. This performance is superior to LR(0.65,0.66,0.65), SVM (0.64, 0.63, 0.65), RF (0.65, 0.66, 0.67), and NB (0.61, 0.63, 0.64). Within deep learning based methods, BiLSTM with FastText embeddings provides the most consistent results, achieving 0.69, 0.69, and 0.70 across Subtasks 1A, 1B, and 1C. This slightly exceeds the performance of CNN (0.68, 0.67, 0.69) and CNN+BiLSTM (0.68, 0.68, 0.69). Nevertheless, all DL models still fall short compared to transformer-based approaches. Among transformers, Bangla-BERT delivers the best results, with micro *F1*-scores of 0.72, 0.72, and 0.73, outperforming m-BERT (0.70, 0.70, 0.69) and XLM-R (0.69, 0.70, 0.71). Finally, the ensemble strategies proved most effective overall. The Weighted Ensemble achieves the highest scores of 0.734, 0.7315, and 0.735, surpassing both individual models and the Averaging Ensemble (0.71, 0.71, 0.72). A key finding of this study is the effectiveness of ensemble strategies. These results show that the weighted ensemble outperforms standalone ML, DL, and transformer models.

6 Error Analysis

Error analysis is performed using both quantitative and qualitative approaches. Detailed results are provided in Appendices A and B. Quantitative analysis identifies systematic misclassifications via confusion matrices, while qualitative analysis explores underlying causes such as class imbalance, contextual subtleties, and overlapping linguistic cues.

7 Conclusion

In this paper, we propose a weighted ensemble approach for multi-task hate speech detection in a Bengali corpus, leveraging the complementary strengths of multiple transformer-based models. By combining fine-tuned Bangla-BERT, m-BERT, and XLM-R, the ensemble captured both language-specific nuances and cross-lingual semantic information, outperforming individual models across all subtasks. It achieved micro-*F1* scores of 0.734, 0.7315, and 0.735 on Subtask 1A, 1B, and 1C, respectively.

To validate our approach, we conduct extensive experiments with traditional machine learning and deep learning models using various feature extraction and embedding strategies. The results highlight the effectiveness of the ensemble and its potential for low-resource languages, where limited annotated data and linguistic complexity pose significant challenges for automated text classification.

Limitations

Our approach has several potential limitations. Notably, the dataset exhibits substantial class imbalance, which can cause models to overfit the majority classes while underperforming on underrepresented ones. Addressing this limitation requires effective data augmentation techniques, such as SMOTE (Chawla et al., 2002), ADASYN (He et al., 2008), or other strategies to increase the amount of data, which can help balance the dataset and improve model generalization.

References

American Library Association. 2017. Hate speech and hate crime. https://www.ala.org/advocacy/intfreedom/hate. Accessed September 22, 2025.

MM Amin. 2024. Ai-powered personalized marketing: A deep dive into customer segmentation and targeting. *INTERNATIONAL JOURNAL*, 11(12).

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.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

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.

Amit Kumar Das, Abdullah Al Asif, Anik Paul, and Md Nur Hossain. 2021. Bangla hate speech detection on social media using attention-based recurrent neural network. *Journal of Intelligent Systems*, 30(1):578–591.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in bengali. *arXiv preprint arXiv:2210.03479*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Koyel Ghosh and Apurbalal Senapati. 2022. Hate speech detection: a comparison of mono and multilingual transformer model with cross-language evaluation. In *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation*, pages 853–865.

Keyan Guo, Alexander Hu, Jaden Mu, Ziheng Shi, Ziming Zhao, Nishant Vishwamitra, and Hongxin Hu. 2023. An investigation of large language models for real-world hate speech detection. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 1568–1573.

Rajan Das Gupta, Md Kishor Morol, Nafiz Fahad, Md Tanzib Hosain, Sumaya Binte Zilani Choya, and Md Jakir Hossen. 2025. Brains: A retrieval-augmented system for alzheimer's detection and monitoring. *arXiv preprint arXiv:2511.02490*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for bengali-english machine translation. *arXiv preprint arXiv:2009.09359*.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee.

Md Tanzib Hosain, Rajan Das Gupta, and Md Kishor Morol. 2025a. Multilingual question answering in low-resource settings: A dzongkha-english benchmark for foundation models. *arXiv preprint arXiv:2505.18638*.

Md Tanzib Hosain and Md Kishor Morol. 2025. B-reaso: A multi-level multi-faceted bengali evaluation suite for foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9260–9274.

Md Tanzib Hosain, Md Kishor Morol, and Md Jakir Hossen. 2025b. A hybrid self attentive linearized phrase structured transformer based rnn for financial sentence analysis with sentence level explainability. *Scientific Reports*, 15(1):23893.

Tanvir Hossan, BM Taslimul Haque, Md Shihabun Sakib, Niladry Chowdhury, and Md Minhajul Amin. 2025. Ethical challenges in business analytics: Balancing data privacy and profit. *Open Access Library Journal*, 12(2):1–12.

Alvi Md Ishmam and Sadia Sharmin. 2019. Hateful speech detection in public facebook pages for the bengali language. In *2019 18th IEEE international conference on machine learning and applications (ICMLA)*, pages 555–560. IEEE.

Md Mainul Islam, Ismoth Zerine, Md Arifur Rahman, Md Saiful Islam, and Md Yousuf Ahmed. 2024. Ai-driven fraud detection in financial transactions-using machine learning and deep learning to detect anomalies and fraudulent activities in banking and e-commerce transactions. *Available at SSRN 5287281*.

Md Saroar Jahan, Mainul Haque, Nabil Arhab, and Mourad Oussalah. 2022. Banglahatebert: Bert for abusive language detection in bengali. In *Proceedings of the second international workshop on resources and techniques for user information in abusive language analysis*, pages 8–15.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Prashant Kapil and Asif Ekbal. 2024. A unified multi-task learning architecture for hate detection leveraging user-based information. *arXiv preprint arXiv:2411.06855*.

Md Rezaul Karim, Sumon Kanti Dey, Tanhim Islam, Sagor Sarker, Mehadi Hasan Menon, Kabir Hossain, Md Azam Hossain, and Stefan Decker. 2021. Deephateexplainer: Explainable hate speech detection in under-resourced bengali language. In *2021 IEEE 8th international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE.

Ashfia Jannat Keya, Md Mohsin Kabir, Nusrat Jahan Shammey, Md Rashedul Islam, and Yutaka Watanobe. 2023. G-bert: an efficient method for identifying hate speech in bengali texts on social media. *IEEE Access*, 11:79697–79709.

Ahmed Cherif Mazari, Nesrine Boudoukhani, and Abdelhamid Djeffal. 2024. Bert-based ensemble learning for multi-aspect hate speech detection. *Cluster Computing*, 27(1):325–339.

Nauros Romim, Mosahed Ahmed, Md Saiful Islam, Arnab Sen Sharma, Hriteshwar Talukder, and Mohammad Ruhul Amin. 2022. Bd-shs: A benchmark dataset for learning to detect online bangla hate speech in different social contexts. *arXiv preprint arXiv:2206.00372*.

Tanjil Hasan Sakib, Md Tanzib Hosain, and Md Kishor Morol. 2025. Small language models: Architectures, techniques, evaluation, problems and future adaptation. *arXiv preprint arXiv:2505.19529*.

Abdus Salam, Ishtiaq Mohammed Chowdhury, Mohammad Masum Sadeque, and BM Taslimul. 2016. Save time for public transport users in a developing country. *International Journal of Education and Management Engineering*, 11(8):27–33.

Hind Saleh, Areej Alhothali, and Kawthar Moria. 2023. Detection of hate speech using bert and hate speech word embedding with deep model. *Applied Artificial Intelligence*, 37(1):2166719.

Tanmay Sen, Ansuman Das, and Mrinmay Sen. 2024. Hatetinyllm : Hate speech detection using tiny large language models. *Preprint*, arXiv:2405.01577.

Kartik Singh, Meenakshi Tripathi, Basant Agarwal, and Abhay Kumar Sain. 2023. Ensemble of transformer based approach for hate speech detection on twitter data. In *2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, volume 10, pages 894–899. IEEE.

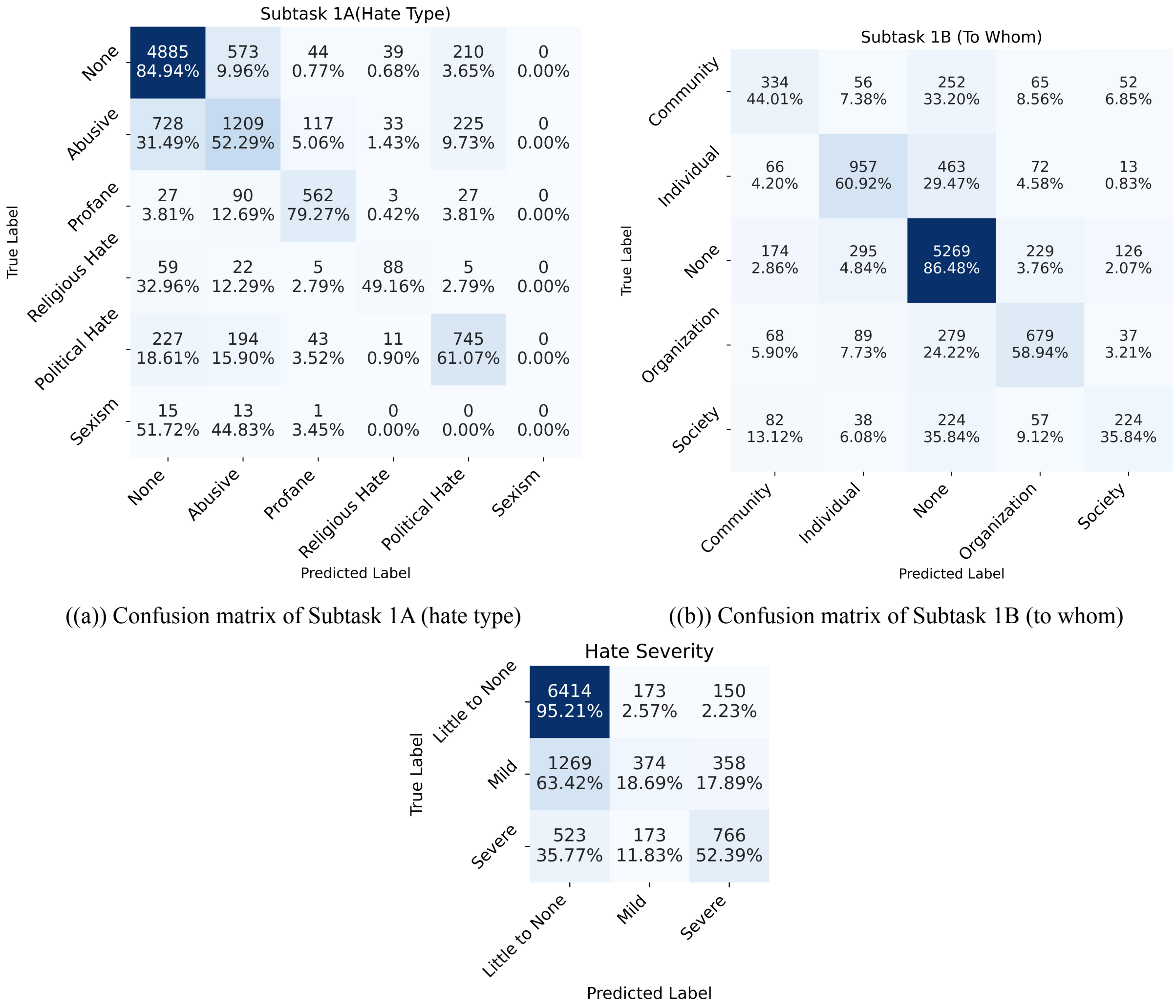
Md Tariquzzaman, Md Wasif Kader, Audwit Anam, Naimul Haque, Mohsinul Kabir, Hasan Mahmud, and Md Kamrul Hasan. 2023. the_linguists at blp-2023 task 1: A novel informal bangla fasttext embedding for violence inciting text detection. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 214–219.

Nishat Vasker, Anika Tabassum Nafisa, MD Arifur Rahman, and Mahamudul Hasan. 2024. Heart disease classification with xai and kernel shap. In *2024 IEEE 3rd International Conference on Robotics, Automation, Artificial-Intelligence and Internet-of-Things (RAAICON)*, pages 82–85. IEEE.

Ismoth Zerine, Md Mainul Islam, Md Saiful Islam, Md Yousuf Ahmad, and Md Arifur Rahman. 2020. Climate risk analytics for us agriculture sustainability: Modeling climate impact on crop yields and supply chain to support federal policies food security and renewable anergy adoption. *Cuestiones de Fisioterapia*, 49(3):241–258.

A Quantitative analysis

In Subtask 1A (hate type), the confusion matrix 3(a) shows that the model performed strongly on the *None* category, achieving an accuracy of 84.94%. However, this dominance contributes to frequent misclassification of the minority hate types. A significant portion of *Abusive* samples are mislabeled as *None* (31.49%, 728 instances) or as *Profane* (5.06%, 117 instances), while *Profane* instances are often predicted as *Abusive* (12.69%, 90 instances). This pattern highlights the fine-grained overlap between abusive language and profanity. Similarly, *Political Hate* is frequently misclassified as *Abusive* (15.90%, 194 instances), reflecting the difficulty of detecting implicit or coded political language. The weakest performance is observed in *Sexism*, where most sexist utterances are mislabeled as *None* (51.72%, 15 instances) or *Abusive* (44.83%, 13 instances), emphasizing the impact of severe class imbalance and subtle linguistic cues.



((a)) Confusion matrix of Subtask 1A (hate type)

((b)) Confusion matrix of Subtask 1B (to whom)

((c)) Confusion matrix of hate severity

Figure 3: Confusion matrix of the proposed ensemble transformer models on task

Text	Hate Type		To Whom		Severity	
	A	P	A	P	A	P
হামাস আর বোম মার (Hamas, bomb them)	Abusive	None	Organization	None	Severe	Mild
সবগুলোই চুর হারামি টাকা খায় (They all take stolen, illicit money)	Profane	Abusive	Community	Individual	Little to None	Severe
আমেরিকা নেটো যুদ্ধ অপরাধী (America and NATO are war criminals)	Political Hate	Abusive	Organization	Society	Little to None	Mild
আজ কাল পবিত্র সংসদ বড়ে গেছে নরতকি আর বেসসা দিয়ে (These days the sacred parliament has become full with prostitutes and immoral people)	Profane	Abusive	Community	Organization	Severe	Mild
শালা মিথ্যুক দালাল ও মুনাফিক (Damn liar, broker, and hypocrite)	Abusive	Profane	None	Individual	Mild	Severe

Table 5: Illustrative data samples highlighting the diverse behavior of the ensemble model. Here, A denotes the ground-truth label, while P denotes the predicted label.

The confusion matrix of Subtask 1B (To Whom) 3(b), non-targeted content (*None*) is identified with relatively high accuracy (86.48%, 5,269 instances). Nevertheless, the majority class again overshadows the minority categories. For example, labels such as *Community*, *Organization*, and *Society* are frequently misclassified, likely due to their overlapping meanings and nuances in the Bengali context. Overall, all labels exhibit a tendency to be predicted as *None*, reflecting the dominance of this category in the dataset.

Regarding the confusion matrix of hate severity 3(c) ,the model tends to underestimate intensity, with a large portion of instances being classified as *Little to None*. Specifically, 63.42% of *Mild* cases and 35.77% of *Severe* cases are predicted as *Little to None*, indicating challenges in distinguishing subtle variations in hate severity.

B Qualitative analysis

The model shows systematic biases and confusions that stem primarily from class imbalance, subtle contextual cues, and overreliance on explicit lexical signals as shown in Table 5. For hate type, the model struggles particularly with *Sexism*, *Religious Hate*, often misclassifying them as *None* or *Abusive*. The dataset does not provide enough representative examples for the model to learn their patterns. In contrast, *Profane* speech is detected more reliably, since it is usually tied to explicit keywords. Common misclassifications include *Abusive* vs. Non-abusive, *Profane* vs. *Abusive*, and *Political Hate* vs. *Abusive*, which arise from overlapping language patterns and insufficient contextual representation. For hate severity, the model frequently mispredicts *Mild*, often confusing it with *Little to None* because of subtle gradations of severity. For Target (to whom), the model distinguishes *Individuals* fairly well but struggles with *Community*, *Society*, and *Organization*. These categories are often implicit, underrepresented, or context-dependent, causing the model to confuse them with one another. The model tends to overpredict majority classes like *None* and *Little to None*, reflecting its bias toward heavily represented categories, while underperforming on minority classes. Moreover, contextual nuance is crucial to separate closely related categories such as *Little to None* vs. *Mild* or *Organization* vs. *Society*.

The ensemble approach helps by boosting confidence and compensating for some data gaps, but categories like *Sexism* remain difficult to predict simply because there is not enough training data to establish strong patterns.

Computational Story Lab at BLP-2025 Task 1:
HateSense: A Multi-Task Learning Framework for Comprehensive Hate Speech Identification using LLMs

Tabia Tanzin Prama[1,2,3,5], **Christopher M. Danforth**[1,2,3,4], **Peter Sheridan Dodds**[1,2,3,5,6]
[1]Computational Story Lab, [2]Vermont Complex Systems Institute,
[3]Vermont Advanced Computing Center, [4]Department of Mathematics and Statistics,
[5]Department of Computer Science, University of Vermont, Burlington, VT 05405, USA
[6]Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501, USA

Abstract

This paper describes HateSense, our multi-task learning framework for the BLP 2025 shared task 1 on Bangla hate speech identification. The task requires not only detecting hate speech but also classifying its type, target, and severity. HateSense integrates binary and multi-label classifiers using both encoder- and decoder-based large language models (LLMs). We experimented with pre-trained encoder models (Bert based models), and decoder models like GPT-4.0, LLaMA 3.1 8B, and Gemma-2 9B. To address challenges such as class imbalance and the linguistic complexity of Bangla, we employed techniques like focal loss and odds ratio preference optimization (ORPO). Experimental results demonstrated that the pre-trained encoders (BanglaBert) achieved state-of-the-art performance. Among different prompting strategies, chain-of-thought (CoT) combined with few-shot prompting proved most effective. Following the HateSense framework, our system attained competitive micro-F1 scores: 0.741 (Task 1A), 0.724 (Task 1B), and 0.7233 (Task 1C). These findings affirm the effectiveness of transformer-based architectures for Bangla hate speech detection and suggest promising avenues for multi-task learning in low-resource languages.

Warning: this paper contains content that may be offensive or upsetting

1 Introduction

The ever-expanding digital landscape, while promising to foster global connectivity and social cohesion, has simultaneously emerged as a breeding ground for hate speech (Castaño-Pulgarín et al., 2021). Hate speech is broadly defined as language that targets, attacks, or incites implicit or explicit hatred or violence against individuals or groups based on specific attributes such as physical appearance, religion, ethnic origin, or gender identity (Papcunová et al., 2021). Its pervasive presence poses severe risks, including the promotion of social division, deterioration of mental health, and escalation of violence (Sahoo et al., 2024). Inadequate moderation of such content further cultivates intolerance, amplifying its negative societal impacts (Hangartner et al., 2021). Addressing online hate speech requires moving beyond a simple "toxic vs. non-toxic" label toward a nuanced analysis that captures its type, severity, and target, enabling deeper insights into motives and potential consequences.

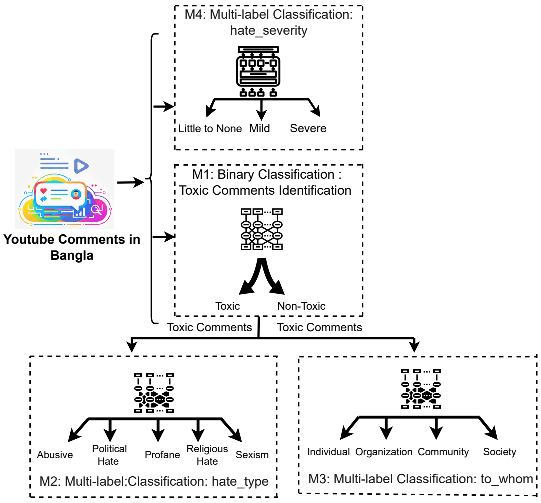

M4: Multi-label Classification: hate_severity
Little to None
Mild
Severe
Youtube Comments in Bangla
M1: Binary Classification : Toxic Comments Identification
Toxic
Non-Toxic
Toxic Comments
Toxic Comments
Abusive
Political Hate
Profane
Religious Hate
Sexism
M2: Multi-label:Classification: hate_type
Individual
Organization
Community
Society
M3: Multi-label Classification: to_whom

Figure 1: The proposed HateSense framework. It begins with M1 (binary hate speech detection), followed by M2 (multi-label classification of hate type, Task 1A) and M3 (multi-label classification of target, Task 1B). A separate model, M4, classifies hate speech severity. The combined outputs of M1–M4 form the results for Task 1C.

The shared task 1 focuses on Bangla multi-task hate speech identification. Our team, under the name Computational StoryLab and username ttprama, participated in the multi-task hate speech

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 430–442
December 23, 2025 ©2025 Association for Computational Linguistics

identification track. For this task, we proposed HateSense shows in Figure 1, a multi-task learning framework designed to classify Bangla texts into predefined categories of hate type, severity, and targeted group. Transformer-based architectures (Vaswani et al., 2017), such as BERT (Devlin et al., 2019), have revolutionized NLP tasks and consistently achieved state-of-the-art (SOTA) performance across benchmarks (Lan et al., 2019). Meanwhile, the recent surge in LLMs has established them as strong candidates for hate speech detection (Liu et al., 2019), particularly in zero-shot settings. However, while significant progress has been made for English and other high-resource languages, research on Bangla—a low-resource language—remains limited. Prior studies have explored Bangla hate speech detection (Jahan et al., 2019; Prama et al., 2025) and LLM-based methods (Shibli et al., 2022), but no prior work has addressed their intersection within a multi-task learning setting.

To address this research gap, our key contributions are as follows:

- We propose a multi-task learning framework, HateSense, which goes beyond binary detection to jointly predict the type, severity, and targeted group of hate speech.
- We establish several encoder-based baselines for each subtask, with encoders further fine-tuned on Bangla achieving state-of-the-art performance on our dataset.
- We investigate zero-shot(ZS), few-shot and COT prompting with state-of-the-art LLMs and introduce a novel translation-based prompting strategy, which outperforms existing methods on our dataset.

2 Task and Data

The dataset consists of Bangla YouTube comments (Hasan et al., 2025a), which serve as the input for all subtasks in Task 1 (Hasan et al., 2025b). The BLP Workshop offers three subtasks: Task 1A: categorizing the type of hate speech (abusive, sexism, religious hate, political hate, profane, or none), Task 1B: identifying the targeted group (individuals, organizations, communities, or society), and Task 1C: is a multi-task setup classifying the severity of hate speech (Little to None, Mild, or Severe), type of hate (Task 1A) and targeted group (Task 1B) in Bangla commentd. The label distributions for each subtask are presented in Tables 1, 2, and 3 respectively.

Label	Train	Dev	Test	Total
None	19954	2898	5751	28603
Abusive	8212	1113	2312	11637
Political Hate	4227	574	1220	6021
Profane	2331	342	709	3382
Religious Hate	676	78	179	933
Sexism	122	19	29	170

Table 1: Label distribution of type of hate speech.

Label	Train	Dev	Test	Total
None	21190	3064	6093	30347
Individual	5646	755	1571	7972
Organization	3846	584	1152	5582
Community	2635	338	759	3732
Society	2205	283	625	3113

Table 2: Label distribution of target group of hate speech.

Label	Train	Dev	Test	Total
Little to None	23489	3417	6737	33643
Mild	6853	909	2001	9763
Severe	5180	698	1462	7340

Table 3: Label distribution of severity of hate speech.

3 Methodology

As Task 1 follows a multi-task setup, we adopted the proposed HateSense framework (Figure 1). Tasks 1A and 1B were performed in two stages: first, binary classification of hate speech (M1), followed by multi-label classification of the predicted hate speech instances (M2 for Task 1A and M3 for Task 1B). Task 1C was addressed using a dedicated multi-label classification model (M4), combined with the outputs of M1, M2, and M3. Our baselines fall into two categories: (1) fine-tuning pretrained language models (LMs) on our dataset, and (2) prompting LLMs with zero-shot (ZS), few-shot (FS), and chain-of-thought (COT) strategies. During evaluation, baseline models were trained on the training set, with the development set used to select the best-performing models. The selected models were then re-trained on the combined training and development sets, and their predictions were submitted for final test set evaluation

3.1 LM Fine-tuning

For fine-tuning, we experiment with several Transformer encoder-based models, including BanglaBERT (Bhattacharjee et al., 2021), BanglaHateBERT (Jahan et al., 2022) , IndicBERT (Bhattacharyya et al., 2023), XLM-Roberta (Conneau et al., 2019), mDistilBERT (Sanh et al., 2019), and mBERT (Devlin et al., 2019). Each subtask is formulated as a multi-label classification problem by adding a classification head on top of the encoder. Given an input sentence

$$S = (w_1, w_2, \ldots, w_n),$$

it is tokenized and passed through a Transformer encoder f_θ, producing contextual representations:

$$H = \{h_1^l, h_2^l, \ldots, h_n^l\}, \quad l \in \{1, \ldots, L\}, \; h_i^l \in \mathbb{R}^d.$$

From the final layer, the hidden state of the special [CLS] token serves as a sentence-level representation:

$$h_{\text{CLS}} = h_1^L \in \mathbb{R}^d.$$

A dropout layer is applied:

$$\tilde{h}_{\text{CLS}} = \text{Dropout}(h_{\text{CLS}}, p = 0.3)$$

and the result is passed through a linear classifier:

$$z = W\tilde{h}_{\text{CLS}} + b, \quad W \in \mathbb{R}^{k \times d}, \; b \in \mathbb{R}^k,$$

where k is the number of labels. Probabilities are obtained via a sigmoid:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}.$$

The model is trained using Binary Cross-Entropy loss:

$$\mathcal{L} = -\frac{1}{k} \sum_{j=1}^{k} \Big[y_j \log(\hat{y}_j) + (1 - y_j) \log(1 - \hat{y}_j) \Big],$$

where $y_j \in \{0, 1\}$. We optimized with AdamW 4×10^{-5} learning rate) for 5 epochs using weighted loss to handle class imbalance, selecting the best checkpoint by validation performance.

3.2 Prompting Strategy

For decoder-only models, each subtask is formulated as a text generation problem, where the model is prompted to produce exactly one label from the predefined set of choices. We experiment with GPT-4.0 (Achiam et al., 2023), LLaMA 3.1 8B (Dubey et al., 2024), and Gemma-2 9B (Riviere et al., 2024). We design base prompts separately for binary classification and multi-label classification. We explore several prompting strategies like Zero-shot prompting (Radford et al., 2019), Few-shot prompting (Brown et al., 2020), Chain-of-Thought (CoT) prompting (Wei et al., 2022)("Let's think step by step" is appended to encourage structured reasoning) and CoT + Few-shot prompting. Appendix A.1, Table 8 , 9, 10 and 11 shows the four prompt strategy we used for this analysis.

3.3 Evaluation Metrics

We evaluate models with micro-F1, aggregating counts over all classes. Let TP_c, FP_c, and FN_c denote true positives, false positives, and false negatives for class $c \in \mathcal{C}$. The micro-F1 is their harmonic mean of micro-precision and micro-recall:

$$\text{F1}_\mu = \frac{2 \sum_c \text{TP}_c}{2 \sum_c \text{TP}_c + \sum_c \text{FP}_c + \sum_c \text{FN}_c}.$$

It weights each instance equally, providing a overall score under class imbalance and across heterogeneous label frequencies.

4 Results and Discussion

4.1 Evaluation Phase

Model	M1	M2	M3	M4
BanglaBERT	**0.865**	**0.741**	**0.724**	**0.754**
BanglaHate BERT	0.809	0.724	0.688	0.682
IndicBERT	0.845	0.694	0.705	0.724
XLM-Roberta	0.843	0.728	0.708	0.736
mDistilBERT	**0.865**	0.709	0.675	0.675
mBERT	0.849	0.704	0.708	0.735
VAC-BERT	0.841	0.687	0.698	0.695

Table 4: Performance (micro F1 scores) of fine-tuned models across hate speech detection (M1), hate speech type (M2), target (M3), and severity (M4).

During the evaluation phase on the development set, we assessed models on Sub-tasks 1A, 1B, and

Model	**M1**	**M2**	**M3**	**M4**
GPT-4o$_{ZS}$	0.701	0.361	0.489	0.453
LLaMA-3.1$_{ZS}$	0.568	0.241	0.336	0.281
Gemma-2$_{ZS}$	0.620	0.226	0.398	0.340
GPT-4o$_{FS}$	0.713	0.**395**	0.478	0.453
LLaMA-3.1$_{FS}$	0.555	0.239	0.319	**0.487**
Gemma-2$_{FS}$	0.607	0.250	0.370	0.326
GPT-4o$_{COT}$	0.736	0.360	0.501	0.476
LLaMA-3.1$_{COT}$	0.579	0.256	0.346	0.298
Gemma-2$_{COT}$	0.628	0.260	0.395	0.357
GPT-4o$_{COT+FS}$	**0.745**	0.365	**0.510**	0.463
LLaMA$_{COT+FS}$	0.593	0.284	0.342	0.301
Gemma-2$_{COT+FS}$	0.638	0.247	0.401	0.349

Table 5: Performance (micro F1 scores) of GPT-4o, LLaMA-3.1, and Gemma-2 under different prompting strategies (ZS = Zero-Shot, FS = Few-Shot, COT = Chain-of-Thought) across hate speech detection (M1), type (M2), target (M3), and severity (M4) classification.

1C. For Sub-tasks 1A and 1B, we employed a two-stage pipeline consisting of a binary toxic comment detector (M1) followed by multi-label classifiers (M2 for hate type and M3 for target). For Sub-task 1C, a separate multi-label classifier (M4) was used in combination with the outputs of M1, M2, and M3.

In toxic comment detection (M1), BanglaBERT and mBERT were the top performers with identical F1 scores of 0.865. Other models, including recent LLMs like GPT-4o$_{COT+FS}$, also demonstrated competitive performance, indicating that both fine-tuned BERT models and large generative models are effective for this task. For Sub-task 1A, performance varied more due to the task's complexity. BanglaBERT achieved the highest F1 score of 0.741, outperforming other fine-tuned models like BanglaHateBERT (0.724) and XLM-Roberta (0.728). Decoder-only LLMs generally struggled with the expanded label space. In Sub-task 1B, BanglaBERT again secured the best performance with an F1 score of 0.7247. The difficulty of this task was evident as only GPT-4o surpassed an F1 score of 0.6 among LLMs. For severity classifier(M4), BanglaBERT was the top fine-tuned model (F1 = 0.7577), while GPT-4o$_{COT+FS}$ led among decoder-only models. Across all subtasks, BanglaBERT consistently delivered the strongest and most reliable performance. Among LLMs, CoT with few-shot prompting proved most effective. Following the HateSense framework, the combined multi-task evaluation for Task 1C (classifying type of hate, severity, and targeted group) in the development phase achieved F1 = 0.7233 (accuracy = 0.7233, precision = 0.7165, recall = 0.7233).

4.2 Testing Phase

Evaluation score	**M1**	**M2**	**M3**	**M4**
F1 score	0.833	0.704	0.703	0.745
precision	0.836	0.701	0.70	0.741
recall	0.832	0.717	0.708	0.765

Table 6: Performance (F1 micro scores) in testing phase of BanglaBERT across hate speech detection (M1), type, target (M3) and severity (M4) classification.

During testing, we retrained our best-performing model from the evaluation phase, BanglaBERT, using the combined training and development sets to obtain a more generalizable classifier. Following the proposed HateSense framework, we trained separate models for each subtask. The toxic comment detector (M1), formulated as a binary classification task (toxic vs. non-toxic), achieved a micro-F1 score of 0.833. Using M1's predictions as an additional input signal, we then trained a multiclass hate-type classifier (M2) for Subtask 1A, which attained a micro-F1 of 0.704. For Subtask 1B, the target-group multiclass classifier (M3) achieved a micro-F1 of 0.703, while the hate severity multiclass classifier (M4) reached a micro-F1 of 0.745. These results are summarized in Table 6. Aggregating across all subtasks in the test set, our HateSense framework attains a micro-F1 of 0.717 for Task 1C (accuracy = 0.717, precision = 0.718, recall = 0.717).

4.3 Error Analysis

Effect of Class Imbalance. Class-wise accuracies (Appendix A.2, Figure 2) and confusion matrices (Figure 3) reveal consistent patterns across subtasks. Due to class imbalance, performance drops notably for minority and nuanced classes like Sexism (Task 1A), Society (Task 1B), and Mild (Task 1C). While the model handles clear categories well, it struggles with subtle hate. Task 1A accuracy is high for None (0.89) and Profane (0.68) but drops to 0.00 for Sexism. Task 1B favors

Comment	**Gold Label**	**Model Prediction**	**Subtask**
তোমার মতন মেয়ে বেঁচে থাকার চেয়ে মরে যাওয়া অনেক ভালো তোমরা হচ্ছ পাপী অন্য ছেলের সাথে ফটো দাও কেন *“For a girl like you, it’s much better to die than to live. You are sinners – why do you post photos with other men?”*	Sexism	Abusive	1A
আগে ইসরাইল বদ করতে হবে সবাই একসাথে *“We must destroy Israel first, all together.”*	Abusive	Political Hate	1A
ইহুদির বাচ্চা ইহুদী ই হবে *“A Jew’s child will be a Jew.”*	Community	Society	1B
ভারতীয় দালাল সময় টিভিকে বয়কট করুন *“Boycott Somoy TV, the Indian stooges.”*	Organization	Community	1B
ইজরায়েলের বিচার হওয়া উচিৎ *“Israel should be brought to justice.”*	Abusive, Mild, Society	Abusive, Little-to-None, Society	1C
আল্লাহ্ এসব জানোয়ারদের শেষ করে দাও *“Allah, wipe out these animals.”*	Profane, Severe, Community	Abusive, Little-to-None, Community	1C

Table 7: Examples of comments, gold labels, and model predictions under the HateSense framework for subtasks 1A (hate type), 1B (target), and 1C (severity). Incorrect model predictions are highlighted in red.

None (0.85) over Society (0.34), and Task 1C excels on Little-to-None (0.95) while failing on Mild (0.36). Confusion matrices confirm these trends: M1 effectively separates toxic from non-toxic, whereas M2–M4 bias toward dominant labels (e.g., None/Little-to-None). Consequently, nuanced hate often misclassifies as neutral; Sexism is never identified (0 correct), and Mild/Severe cases are frequently predicted as Little-to-None.

Qualitative Error Analysis. Table 7 illustrates characteristic failures across subtasks. In Subtask 1A (type classification), we observe frequent confusion between Abusive and Profane, alongside under-predictions of subtle Political Hate, often stemming from short texts or figurative language. For Subtask 1B (target identification), the primary challenge lies in distinguishing Organization versus Community, particularly when targets are implied or indirect. Despite multitask modeling capturing interdependencies, systematic errors persist, including specificity loss, target mismatches, and severity underestimation:

- **Sexism → Generic Abuse:** Explicit gendered hate (e.g., তোমার মতন মেয়ে বেঁচে..) is reduced to generic *Abusive*, missing honor-policing and sexism.
- **Target Granularity Confusion:** The model detects group hate but confuses labels (e.g., Jews → *Society*, TV channels → *Community*), failing to distinguish Community / Society / Organization.
- **Severity Underestimation:** Violent rhetoric (e.g., “আল্লাহ্ এসব জানোয়ারদের..”) is gold-labeled *Severe* but predicted as *Little-to-None*, showing the model downgrades threat intensity.

5 Conclusion

We introduced HateSense, a multi-task framework for Bangla hate speech detection that jointly models hate type, target, and severity. Leveraging encoder–decoder transformers with focal loss, Odds Ratio Preference Optimization (ORPO), and CoT + few-shot prompting, our system achieved strong performance across all subtasks in BLP 2025, demonstrating the effectiveness of transformer-based approaches for low-resource, morphologically rich languages. At the same time, our analysis exposed persistent challenges, particularly class imbalance and the difficulty of modeling underrepresented categories such as Sexism and Religious Hate. In future work, we plan to explore data augmentation, cross-lingual transfer, and more robust multitask architectures to improve fine-grained Bangla hate speech detection and extend these methods to other low-resource languages. Our code is available for reproducibility at: https://github.com/HateSense.

6 Limitations

While our framework achieved strong performance, several limitations remain. First, fine-tuning decoder-based models such as GPT, LLaMA, or Gemma on the shared task dataset could further improve performance; however, computational resource constraints prevented us from exploring this option. Second, the dataset suffers from class imbalance. For example, hate speech type Sexism, target category Society, and severity level Severe are underrepresented, leading to reduced model performance in these classes. Although we employed focal loss and Odds Ratio Preference Optimization (ORPO) to address imbalance, the models still struggle with fine-grained distinctions in ambiguous or borderline cases.

Moreover, as Bangla is a low-resource language, language models face challenges in capturing cultural-specific hate speech phenomena, which limits their ability to generalize beyond surface-level patterns. Fine-tuning on more culturally nuanced datasets could enhance detection accuracy. Another constraint is the limited availability of pre-trained models specifically focused on Bangla, which restricts opportunities for leveraging domain-specific linguistic features. Finally, the overall dataset size for training and evaluation remains relatively small, which reduces the robustness and generalizability of our models to real-world applications.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, and 260 others. 2023. Gpt-4 technical report.

Abhik Bhattacharjee, Tahmid Hasan, Kazi Samin Mubasshir, Md. Saiful Islam, Wasi Uddin Ahmad, Anindya Iqbal, M. Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. In *NAACL-HLT*.

Pramit Bhattacharyya, Joydeep Mondal, Subhadip Maji, and Arnab Bhattacharya. 2023. Vacaspati: A diverse corpus of bangla literature. *ArXiv*, abs/2307.05083.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Sergio Andrés Castaño-Pulgarín, Natalia Suárez-Betancur, Luz Magnolia Tilano Vega, and Harvey Mauricio Herrera López. 2021. Internet, social media and online hate speech. systematic review. *Aggression and Violent Behavior*, 58:101608.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *ArXiv*, abs/1911.02116.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Dominik Hangartner, Gloria Gennaro, Sary Alasiri, Nicholas Bahrich, Alexandra Bornhoft, Joseph Boucher, Buket Buse Demirci, Lauren M. Derksen, Aldo Hall, Matthias Jochum, María M. Muñoz, Marc Richter, Franziska Vogel, Salome Wittwer, Felix Wüthrich, Fabrizio Gilardi, and Karsten Donnay. 2021. Empathy-based counterspeech can reduce racist hate speech in a social media field experiment. *Proceedings of the National Academy of Sciences of the United States of America*, 118.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Maliha Jahan, Istiak Ahamed, Md. Rayanuzzaman Bishwas, and Swakkhar Shatabda. 2019. Abusive comments detection in bangla-english code-mixed and transliterated text. *2019 2nd International Conference on Innovation in Engineering and Technology (ICIET)*, pages 1–6.

Md Saroar Jahan, Mainul Haque, Nabil Arhab, and Mourad Oussalah. 2022. Banglahatebert: Bert for abusive language detection in bengali. In *RESTUP*.

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*, abs/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*, abs/1907.11692.

Jana Papcunová, Marcel Martonik, Denisa Fedáková, Michal Kento, Miroslava Bozogáňová, Ivan Srba, Róbert Móro, Matú Pikuliak, Marián Simko, and Matú Adamkovi. 2021. Hate speech operationalization: a preliminary examination of hate speech indicators and their structure. *Complex & Intelligent Systems*, 9:2827–2842.

Tabia Tanzin Prama, Jannatul Ferdaws Amrin, Md. Mushfique Anwar, and Iqbal H. Sarker. 2025. Ai enabled user-specific cyberbullying severity detection with explainability. *ArXiv*, abs/2503.10650.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Gemma Team Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L'eonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram'e, Johan Ferret, Peter Liu, Pouya Dehghani Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, and 176 others. 2024. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118.

Nihar Ranjan Sahoo, Gyana Prakash Beria, and Pushpak Bhattacharyya. 2024. Indicconan: A multilingual dataset for combating hate speech in indian context. In *AAAI Conference on Artificial Intelligence*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

G. M. Shahariar Shibli, Md. Tanvir Rouf Shawon, Anik Hassan Nibir, Md. Zabed Miandad, and Nibir Chandra Mandal. 2022. Automatic back transliteration of romanized bengali (banglish) to bengali. *Iran Journal of Computer Science*, 6:69–80.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

A Appendix

A.1 Prompts

Task	**Zero-Shot Prompt**
Binary Hate Speech Detection	You are an expert at detecting hate speech in Bangla text. Your task is to classify each input text as either: - true (if it contains hate speech) - false (if it does not contain hate speech) Text: ”<INSERT_TEXT_HERE>” Answer:
Hate Speech Type Classification	You are an expert at detecting hate speech in Bangla text. Your task is to classify each input text into one of the following categories: - Abusive - Sexism - Religious Hate - Political Hate - Profane - Non-hate Text: ”<INSERT_TEXT_HERE>” Answer:
Target of Hate Speech	You are an expert at analyzing hate speech in Bangla text. Your task is to identify the primary target of hate speech in each input text. The possible targets are: - Individuals - Organizations - Communities - Society - Non-hate Text: ”<INSERT_TEXT_HERE>” Answer:
Hate Speech Severity	You are an expert at detecting hate speech in Bangla text. Your task is to classify the severity of the text into one of the following categories: - Little to None - Mild - Severe Text: ”<INSERT_TEXT_HERE>” Answer:

Table 8: Zero-shot prompts for all four hate speech classification subtasks (Hate Speech Detection (M1), Hate Speech Type Classification (M2), Target of Hate Speech (M3) and Hate Speech Severity (M4).

Task	**Few-Shot Prompt**
Binary Hate Speech Detection	You are an expert at detecting hate speech in Bangla text. Classify each input text as either True (if it contains hate speech) or False (if it does not). Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” Answer: True Text: ”১২ বছরের ভিতর কোনো ভোট দিতে পারিনি” Answer: False Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:
Hate Speech Type Classification	You are an expert at detecting hate speech in Bangla text. Classify each input text into one of the following categories: Abusive, Sexism, Religious Hate, Political Hate, Profane, Non-hate Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” Answer: Abusive Text: ”শামীম ওসামা বিন হাসিনা” Answer: Non-hate Text: ”ইহুদির বাচ্চা ইহুদী ই হবে” Answer: Religious Hate Text: ”যে রাষ্ট্র আমেরিকার সাথে নিজেদের তুলনা দেয় হঠাৎ কোনও গরিব হয়ে গেলো এখন ঋণ চায় তার মানে কি চাপাবাজ আওয়ামী লীগ” Answer: Political Hate Text: ”সময় টিভি একটা জাউড়া মিডিয়া মিথ্যা তথ্য প্রচার করে বেড়ায়” Answer: Profane Text: ”তুই মারা গেলে ভালো কারণ তার স্ত্রী থাকতে কেন তার সাথে শুইতে গেলি” Answer: Sexism Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:
Target of Hate Speech	You are an expert at detecting the target of hate speech in Bangla text. Classify each input text into one of: Individuals, Organizations, Communities, Society, Non-hate. Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” Answer: Society Text: ”শামীম ওসামা বিন হাসিনা” Answer: Non-hate Text: ”আল্লাহ্ এসব জানোয়ারদের শেষ করে দাও” Answer: Community Text: ”আলহামদুলিল্লাহ দেশ এগিয়ে যাচ্ছে বিএনপি জামাতীদের জুতা পেটা করতে হবে” Answer: Organization Text: ”চোর চোর ভোট চোর হাসিনা ভোট চোর” Answer: Individual Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:
Hate Speech Severity	You are an expert at assessing the severity of hate speech in Bangla text. Classify each input text into one of: Little to None, Mild, Severe. Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” Answer: Mild Text: ”শামীম ওসামা বিন হাসিনা” Answer: Little to None Text: ”আল্লাহ্ এসব জানোয়ারদের শেষ করে দাও” Answer: Severe Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:

Table 9: Few-shot prompts for the four hate speech classification subtasks (Hate Speech Detection (M1), Hate Speech Type Classification (M2), Target of Hate Speech (M3) and Hate Speech Severity (M4).

Task	**CoT Prompt**
Binary Hate Speech Detection	You are an expert at detecting hate speech in Bangla text. Internally follow these reasoning steps: 1. Read the full sentence. 2. Identify offensive or hostile words/phrases. 3. Consider the context to see if the text expresses hate. 4. If hate elements are present → classify as True. Otherwise → False. Important: Do all reasoning internally and return only the final classification. Text: ”<INSERT_TEXT_HERE>” Answer (True or False):
Hate Speech Type Classification	You are an expert at detecting hate speech in Bangla text. Internally follow these reasoning steps: 1. Read the sentence carefully. 2. Identify abusive, gender-related, religious, political, or profane words. 3. Consider the context to determine the type of hate. 4. Map the text into one of these categories: - Abusive - Sexism - Religious Hate - Political Hate - Profane - Non-hate Important: Do all reasoning internally and return only the final category. Text: ”<INSERT_TEXT_HERE>” Answer:
Target of Hate Speech	You are an expert at analyzing the target of hate speech in Bangla text. Internally follow these reasoning steps: 1. Read the sentence. 2. Identify who/what is being attacked. 3. Determine if the target is an individual, organization, community, or society. 4. If no hate is detected, return Non-hate. Possible categories: - Individual - Organization - Community - Society - Non-hate Important: Do all reasoning internally and return only the final target. Text: ”<INSERT_TEXT_HERE>” Answer:
Hate Speech Severity	You are an expert at detecting the severity of hate speech in Bangla text. Internally follow these reasoning steps: 1. Read the sentence. 2. Identify hostile or violent language. 3. Check if the tone is harmless, mildly offensive, or severely hateful/violent. 4. Classify into one of the following categories: - Little to None - Mild - Severe Important: Do all reasoning internally and return only the final severity label. Text: ”<INSERT_TEXT_HERE>” Answer:

Table 10: Chain-of-Thought (CoT) prompts for the four hate speech classification subtasks (Hate Speech Detection (M1), Hate Speech Type Classification (M2), Target of Hate Speech (M3) and Hate Speech Severity (M4).

Table 11: Chain-of-Thought + Few-shot prompts for the four hate speech classification subtasks (Hate Speech Detection (M1), Hate Speech Type Classification (M2), Target of Hate Speech (M3) and Hate Speech Severity (M4)

Task	CoT + Few-Shot Prompt
Binary Hate Speech Detection	You are an expert at detecting hate speech in Bangla text. Internally follow these steps: 1. Read the sentence fully. 2. Identify offensive or hostile words. 3. Consider the context to see if hate is expressed. 4. Decide: True if hate speech, False otherwise. Important: Do all reasoning internally and return only the final classification. Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” Answer: True Text: ”১২ বছরের ভিতর কোনো ভোট দিতে পারিনি” Answer: False Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:
Hate Speech Type Classification	You are an expert at detecting hate speech in Bangla text. Internally follow these steps: 1. Read the sentence carefully. 2. Identify abusive, gender-related, religious, political, or profane terms. 3. Map them into one category. Categories: Abusive, Sexism, Religious Hate, Political Hate, Profane, Non-hate Important: Do all reasoning internally and return only the final category. Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” → Abusive Text: ”শামীম ওসামা বিন হাসিনা” → Non-hate Text: ”ইহুদির বাচ্চা ইহুদী ই হবে” → Religious Hate Text: ”যে রাষ্ট্র আমেরিকার সাথে নিজেদের তুলনা দেয়…” → Political Hate Text: ”সময় টিভি একটা জাউড়া মিডিয়া…” → Profane Text: ”তুই মারা গেলে ভালো…” → Sexism Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:
Target of Hate Speech	You are an expert at identifying the target of hate speech in Bangla text. Internally follow these steps: 1. Read the sentence carefully. 2. Identify who/what is being attacked. 3. Map the target into one of the given categories. Categories: Individual, Organization, Community, Society, Non-hate Important: Do all reasoning internally and return only the final target. Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” → Society Text: ”শামীম ওসামা বিন হাসিনা” → Non-hate Text: ”আল্লাহ্ এসব জানোয়ারদের শেষ করে দাও” → Community Text: ”আলহামদুলিল্লাহ দেশ এগিয়ে যাচ্ছে বিএনপি জামাতীদের জুতা পেটা করতে হবে” → Organization Text: ”চোর চোর ভোট চোর হাসিনা ভোট চোর” → Individual Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:
Hate Speech Severity	You are an expert at detecting the severity of hate speech in Bangla text. Internally follow these steps: 1. Read the sentence. 2. Identify hostile or violent language. 3. Judge whether it is harmless, mildly offensive, or severely hateful. Categories: Little to None, Mild, Severe Important: Do all reasoning internally and return only the final severity label. Examples: Text: ”ইজরায়েলের বিচার হওয়া উচিৎ” → Mild Text: ”শামীম ওসামা বিন হাসিনা” → Little to None Text: ”আল্লাহ্ এসব জানোয়ারদের শেষ করে দাও” → Severe Now classify the following text: Text: ”<INSERT_TEXT_HERE>” Answer:

A.2 Class-wise perfromance analysis

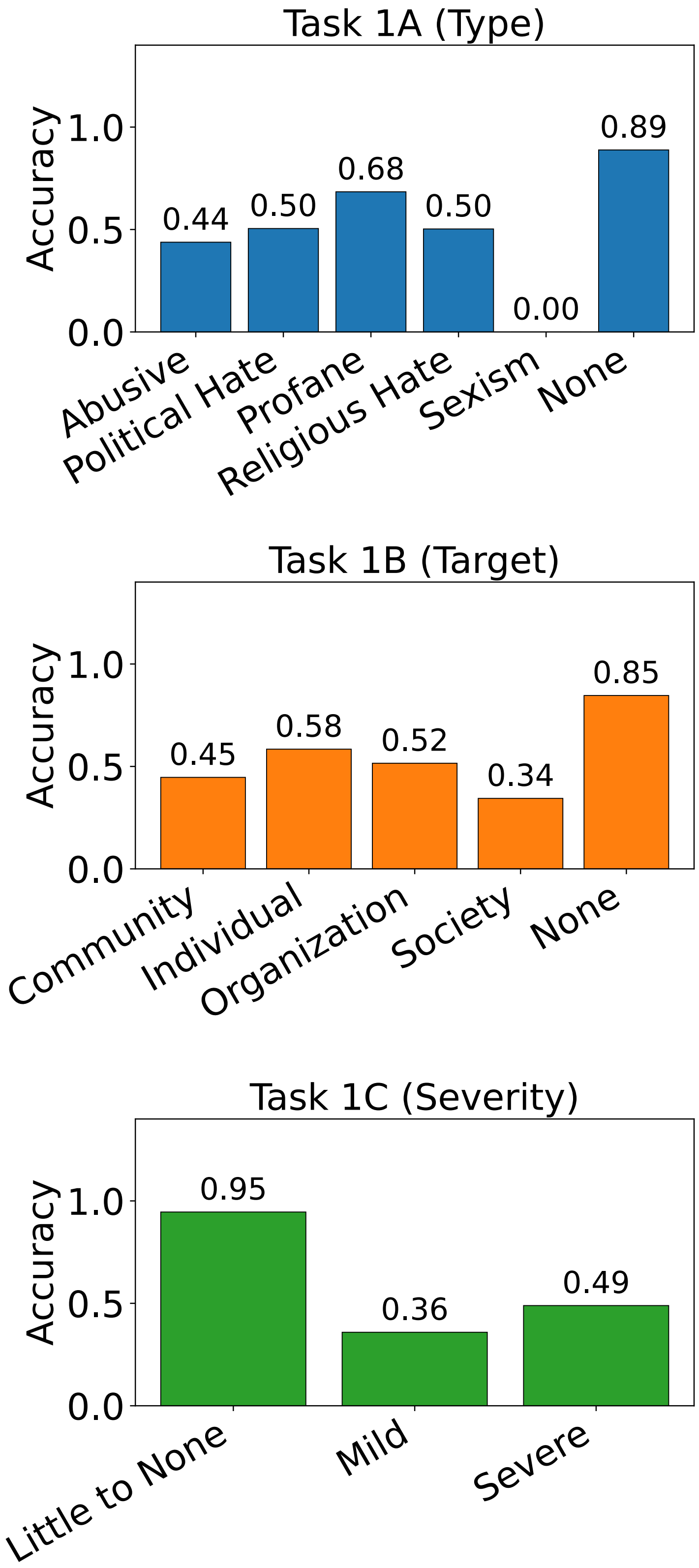
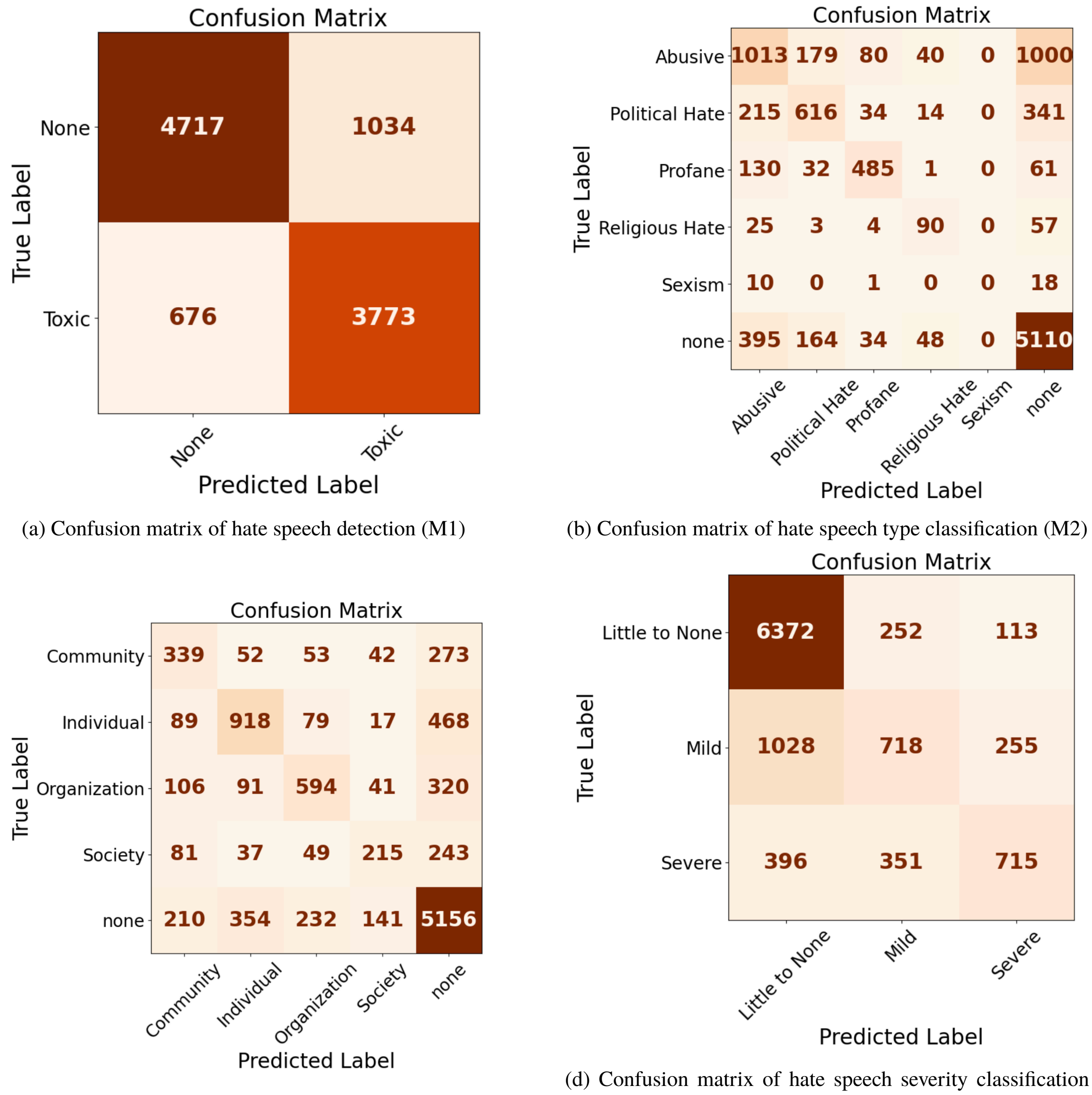


Figure 2: Class-wise accuracy for Task 1A (type), Task 1B (target), and Task 1C (severity).



(a) Confusion matrix of hate speech detection (M1)

(b) Confusion matrix of hate speech type classification (M2)

(c) Confusion matrix of hate speech target classification (M3)

(d) Confusion matrix of hate speech severity classification (M4)

Figure 3: Confusion matrices of the HateSense framework across all models: (a) Hate speech detection(M1), (b) Hate type classification (M2), (c) Hate target classification (M3), (d) Hate severity classification (M4) of the test phase.

CUET-NLP_Zenith at BLP-2025 Task 1: A Multi-Task Ensemble Approach for Detecting Hate Speech in Bengali YouTube Comments

Md. Refaj Hossan, Kawsar Ahmed, and Mohammed Moshiul Hoque
Department of Computer Science and Engineering
Chittagong University of Engineering & Technology, Chittagong 4349, Bangladesh
{u1904007, u1804017}@student.cuet.ac.bd
moshiul_240@cuet.ac.bd

Abstract

Hate speech on social media platforms, particularly in low-resource languages like Bengali, poses a significant challenge due to its nuanced nature and the need to understand its type, severity, and targeted group. To address this, the Bangla Multi-task Hate Speech Identification Shared Task at BLP 2025 adopts a multi-task learning framework that requires systems to classify Bangla YouTube comments across three subtasks simultaneously: type of hate, severity, and targeted group. To tackle these challenges, this work presents **BanTriX**, a transformer ensemble method that leverages BanglaBERT-I, XLM-R, and BanglaBERT-II. Evaluation results show that the **BanTriX**, optimized with cross-entropy loss, achieves the highest weighted micro F1-score of 73.78% in Subtask 1C, securing our team 2$^{\text{nd}}$ place in the shared task.

1 Introduction

Hate speech identification relies on detecting and classifying harmful or offensive language in text, with careful analysis of its type (such as personal attack or communal hate), severity (ranging from mild to severe), and targeted group (including gender and religion); these factors play a critical role in fostering safe online environments (Fayaz et al., 2025). Particularly in low-resource languages (LRLs) like Bengali, the limited availability of annotated datasets and the inherent linguistic complexity present significant challenges. The necessity for multi-task learning, where models must classify multiple related objectives at once and reflect the interconnectedness of real-world scenarios, further complicates this task. The scarcity of comprehensive datasets has hindered progress, the contextual subtlety of Bengali hate speech, and a lack of previous multi-task learning frameworks for LRLs like Bengali. In response, the BLP Workshop@IJCNLP-AACL 2025 organized a shared task (Hasan et al., 2025b) centering on multi-task hate speech identification in Bengali YouTube comments, with classification by type (abusive, religious, or political), severity (mild or severe), and targeted group (society or organization). This collaborative effort highlights the importance of synergy in advancing robust and interpretable hate speech detection systems. Such collaboration forms the central motivation for our work. Our main contributions are summarized as follows:

- We developed **BanTriX**, a robust ensemble that merges BanglaBERT-I, XLM-R, and BanglaBERT-II for multi-task hate speech classification in Bengali.
- By evaluating diverse deep learning, transformer models and their ensembles with comprehensive metrics and ablation studies, we identify the optimal multi-task strategy.
- To enhance interpretability, we employ LIME to highlight feature importance and illuminate the decision processes of our proposed architecture.

2 Related Work

In recent years, researchers have explored harmful online behaviors, e.g., cyberbullying and abusive language, often treating them as related to hate speech. Within this space, automated hate speech detection has progressed rapidly, initially focusing on English datasets (Davidson et al., 2017; Founta et al., 2018), and later expanding to languages such as Arabic (Omar et al., 2020), Spanish (del Arco et al., 2021), and Bengali (Das et al., 2022). This shift was facilitated by shared tasks such as HASOC (Mandl et al., 2025), CHiPSAL (Sarveswaran et al., 2025), and DravidianLangTech@NAACL 2025 (G et al.,

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 443–452
December 23, 2025 ©2025 Association for Computational Linguistics

2025). Several studies have provided a comprehensive overview of hate speech detection techniques, highlighting key contexts (Maruf et al., 2024; Nandi et al., 2024).

A study by Acharya et al. (2025) evaluated FastText and BERT for hate speech detection and target identification, finding that FastText with data augmentation performed best for hate speech (F1 score of 0.8552). At the same time, BERT excelled in target identification (F1 score of 0.5785). Farsi et al. (2024) explored LR, SVM, CNN, XLM-R, and MuRIL, achieving the best result with Indic-SBERT (macro F1 of 0.7013). A FastText model for Hindi offensive text classification achieved 92.2% accuracy on the DHOT dataset (Jha et al., 2020). However, several works addressed aggressive content in Bengali. For instance, Remon et al. (2022) introduced a 10,133-comment Facebook dataset, where SVM with FastText embeddings performed best. Fayaz et al. (2025) proposed BIDWESH, covering regional dialects with 9k+ samples for fair detection. Sharif et al. (2022) presented M-BAD with 15,650 texts for aggression and target detection, achieving a weighted F1 of 0.92 and 0.83 using BanglaBERT. A 30k-comment Bengali dataset from YouTube and Facebook annotated in 7 categories, with SVM reaching 87.5% accuracy (Romim et al., 2020).

Despite significant progress in hate speech and offensive content detection, including multimodal approaches (Hossain et al., 2022; Hee et al., 2023), to the best of our knowledge, no prior studies have addressed a multi-task setup that simultaneously predicts hate type, severity, and targeted group in Bengali. Building on this gap, this study presents a multi-task learning scenario using Bengali text from YouTube comments, aiming to develop robust systems for comprehensive analysis of hate speech.

3 Task and Dataset Description

This study develops a system to classify Bengali YouTube comments by hate type (such as *Abusive*, *Political Hate*, or *Profane*), severity (like *Mild* or *Severe*), and the group targeted (for example, *Individual*, *Organization*, or *Society*). This multi-task approach helps capture how different aspects of hate speech are connected in Bengali. The dataset (Hasan et al., 2025a) contains annotated Bengali comments divided into training, validation, and test sets. For example, the training set has 8,212 *Abusive* and 4,227 *Political Hate* cases, 23,489 with *little* to *no severity*, and 5,646 targeting *individuals*. These figures show the dataset's variety, which supports strong model development. However, there is an imbalance in the dataset, with more samples labeled as *None hate* type (19,954 in training, 1,451 in validation, and 5,751 in test) compared to other categories, as shown in Table 1. Appendix A provides further exploratory data analysis.

Subtask	**Classes**	**Train**	**Valid**	**Test**	**W$_T$**
Hate Type	Abusive	8212	564	2312	153869
	Political Hate	4227	291	1220	109447
	Profane	2331	157	709	43618
	Religious Hate	676	38	179	14659
	Sexism	122	11	29	2396
	None	19954	1451	5751	341607
	Total	**35522**	**2512**	**10200**	**665596**
Hate Severity	Little to None	23489	1703	6737	414639
	Mild	6853	483	2001	146920
	Severe	5180	326	1462	104037
	Total	**35522**	**2512**	**10200**	**665596**
Targeted Group	Individual	5646	364	1571	102771
	Organization	3846	292	1152	84773
	Community	2635	179	759	59800
	Society	2205	141	625	52132
	None	21190	1536	6093	366120
	Total	**35522**	**2512**	**10200**	**665596**

Table 1: Class-wise distribution of datasets used for the task, where W$_T$ denotes total words.

4 System Overview

This section presents the implementation details of the proposed architecture, encompassing both deep learning and transformer-based models.

4.1 Problem Formulation

Given a set of Bangla YouTube comments $C = \{C_1, \ldots, C_{|C|}\}$, each comment C_i is represented by a text sequence X_i. The goal is to learn a mapping f that assigns three labels $Y_i = \{Y_i^{ht}, Y_i^{hs}, Y_i^{tw}\}$ corresponding to hate type (6 classes), hate severity (3 classes), and targeted group (5 classes), formulating a multi-task classification problem: $f : X_i \rightarrow Y_i$. The models performance is governed by a summed cross-entropy loss function $\mathcal{L} = \mathcal{L}_{ht} + \mathcal{L}_{hs} + \mathcal{L}_{tw}$, which quantifies the divergence between predicted and true labels across tasks. To achieve this, the model solves the optimization problem as shown in Eq. 1.

$$\begin{aligned} \min_f \sum_{i=1}^{|C|} \big[& \mathcal{L}_{ht}(f_{ht}(X_i), Y_i^{ht}) \; + \; \mathcal{L}_{hs}(f_{hs}(X_i), Y_i^{hs}) \\ & + \mathcal{L}_{tw}(f_{tw}(X_i), Y_i^{tw}) \big], \end{aligned} \tag{1}$$

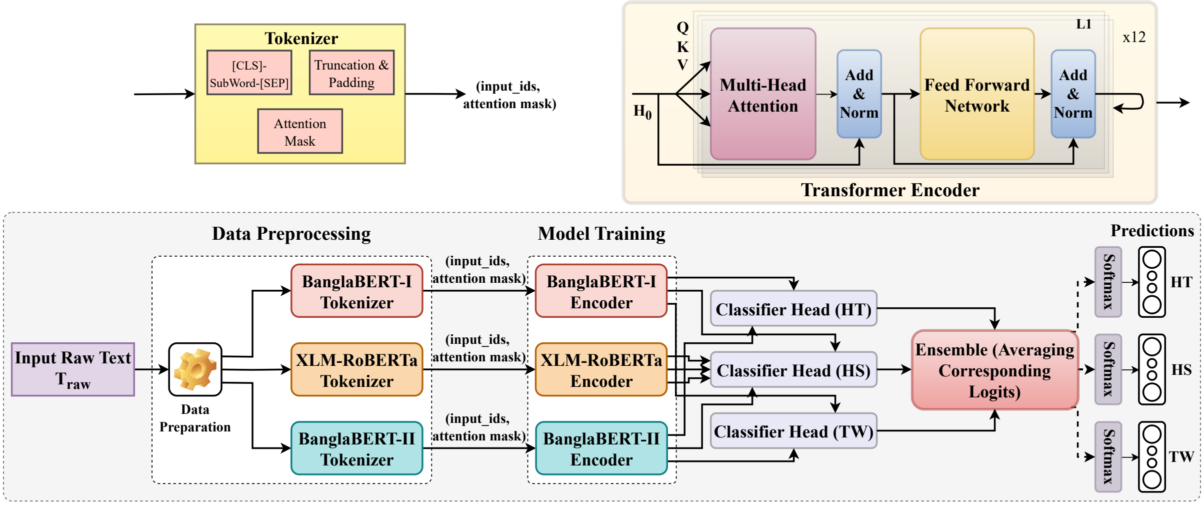


Figure 1: Overview of the proposed architecture for multi-task classification.

where f_{ht}, f_{hs}, f_{tw} are task-specific heads on shared transformer features, and $\mathcal{L}_{ht}, \mathcal{L}_{hs}, \mathcal{L}_{tw}$ are cross-entropy losses for each task.

4.2 Baselines

Several deep learning and transformer-based models were explored to develop the proposed system.

4.2.1 Deep Learning Models

Deep learning models such as CNN (Kim, 2014), BiLSTM (Huang et al., 2015), BiLSTM+CNN, shared a common 128-dimensional word embedding layer trained from scratch on the preprocessed corpus using a `Tokenizer` with <OOV> handling. Input sequences were standardized to 128 tokens through post-padding and truncation. The BiLSTM model stacked two bidirectional LSTM layers (128 units each) with a 10% dropout and three parallel output heads. The CNN model applied two Conv1D layers (128 filters, kernel size 5) with max pooling and global max pooling, followed by dropout and identical output heads. The hybrid model combined a 64-unit BiLSTM branch and a 64-filter CNN branch, merged their outputs, and connected them to the output heads. All models were optimized with Adam using task-specific learning rates (2.15×10^{-4}, 1.25×10^{-3}, and 1.5×10^{-4}, respectively), trained with `SparseCategoricalCrossentropy` loss and early stopping (patience of 13) over up to 25–31 epochs with a batch size of 32.

4.2.2 Transformer-Based Models

Transformer-based models such as BanglaBERT-I (SagorSarker[1]), XLM-R (Conneau et al., 2020), and BanglaBERT-II (Bhattacharjee et al., 2022) were fine-tuned for the task using AdamW with learning rates of 2.25e-5, 2e-5, and 2.35e-5, respectively, over 9 epochs and a batch size of 32. All models processed inputs with a maximum sequence length of 128, applied 10% dropout after the [CLS] pooled output, and shared three linear classification heads for hate type, severity, and target prediction. Training used summed CrossEntropy loss across tasks, with linear learning rate warmup (50 steps for BanglaBERT-I, 175 for BanglaBERT-II, 225 for XLM-R) followed by decay over total training steps. After evaluating transformer-based models, BanglaBERT-I, XLM-RoBERTa, and BanglaBERT-II emerged as the best performers, significantly outperforming mBERT. These three models were then ensembled through systematic exploration to identify the optimal strategy (detailed in Appendix C), using their complementary strengths in contextual understanding to develop **BanTriX**.

4.3 Proposed Approach

Figure 1 illustrates the proposed architecture **(BanTriX)** for Bengali hate speech detection in YouTube comments, integrating a shared transformer backbone with task-specific heads and ensembles of BanglaBERT-I, XLM-R, and BanglaBERT-II. Full implementation details can

[1] `https://huggingface.co/sagorsarker/bangla-bert-base`

be found in the GitHub repository[2].

4.3.1 Data Preparation and Input Layer

The data preparation began with processing raw Bengali text sequences from YouTube comments. Using pre-trained tokenizers (e.g., for BanglaBERT or XLM-R), each text t was converted into input IDs I and attention masks A as depicted in Eq. 2.

$$\{\mathrm{I}, \mathrm{A}\} = \text{tokenizer}(t), \tag{2}$$

where $\mathrm{I}, \mathrm{A} \in \mathbb{R}^{B \times L}$, with batch size $B = 32$ and maximum sequence length $L = 128$.

4.3.2 Shared Encoder

The core of the architecture is a pre-trained transformer encoder (e.g., BanglaBERT-I, XLM-R, or BanglaBERT-II), which extracts contextual embeddings shared across all tasks as shown in Eq. 3.

$$\mathrm{H} = f_{\text{base}}(\mathrm{I}, \mathrm{A}), \tag{3}$$

where $\mathrm{H} \in \mathbb{R}^{B \times L \times D}$ and $D = 768$ is the hidden dimension. This shared encoder captures the linguistic nuances critical for the tasks.

4.3.3 Pooling and Dropout

To create a compact representation, the sequence embeddings were pooled, typically using the `[CLS]` token, yielding $\mathrm{P} \in \mathbb{R}^{B \times D}$. Dropout (0.1) was applied for regularization, as shown in Eq. (4).

$$\mathrm{D} = \text{Dropout}(\mathrm{P}), \tag{4}$$

4.3.4 Task-Specific Heads

From the pooled features D, three linear classifiers were used to produce logits for each task, i.e., hate type ($C_{ht} = 6$), hate severity ($C_{hs} = 3$), and targeted group ($C_{tw} = 5$), as shown in Eq. 5.

$$\begin{aligned} \mathrm{L}_{ht} &= \mathrm{W}_{ht}\mathrm{D} + \mathrm{b}_{ht}, \quad \mathrm{L}_{hs} = \mathrm{W}_{hs}\mathrm{D} + \mathrm{b}_{hs}, \\ \mathrm{L}_{tw} &= \mathrm{W}_{tw}\mathrm{D} + \mathrm{b}_{tw}, \end{aligned} \tag{5}$$

where $\mathrm{L}_{ht} \in \mathbb{R}^{B \times 6}$, $\mathrm{L}_{hs} \in \mathbb{R}^{B \times 3}$, $\mathrm{L}_{tw} \in \mathbb{R}^{B \times 5}$, and W, b are learnable parameters. During inference, probabilities were computed via softmax as $\mathrm{Pr}_{ht} = \text{softmax}(\mathrm{L}_{ht})$, and selected predictions as $\hat{y}_{ht} = \arg\max(\mathrm{Pr}_{ht})$.

4.3.5 Ensemble Mechanism

To enhance performance, three models were ensemble by averaging their logits as shown in Eq. 6.

$$\bar{\mathrm{L}}_{ht} = \frac{1}{3} \sum_{m=1}^{3} \mathrm{L}_{ht}^{(m)}, \tag{6}$$

and similarly for $\bar{\mathrm{L}}_{hs}$ and $\bar{\mathrm{L}}_{tw}$. Final predictions are obtained via softmax on $\bar{\mathrm{L}}$.

4.3.6 System Requirements

The proposed architecture was trained on Kaggle's free-tier environment using two NVIDIA T4 GPUs in a distributed setup, requiring approximately 5 GB of system RAM and ∼15 GB of GPU memory, with a total training time of around 65 minutes. Table B.1 in Appendix B provides the tuned hyperparameters used in the proposed architecture for the tasks.

5 Results and Discussion

Table 2 summarizes the performance of various approaches for the task, with performance metrics including the overall Weighted Micro F1-Score (μ-F1), True Positive Rate (TPR), and Balanced Error Rate (BER). The following insights are drawn from these results.

Which tasks are hard to solve? The three classification tasks present challenges, particularly due to class imbalances. The CNN+BiGRU model shows a high BER for Hate Type (56.06%), indicating difficulty in correctly classifying all classes. Even transformer-based models like BanglaBERT-II exhibit elevated BER for Hate Type (47.41%). The proposed ensemble achieves lower BERs (e.g., 41.07% for Hate Severity). Still, these values remain relatively high, highlighting that Hate Type and Targeted Group are tough to classify accurately due to their complex class distributions (clarified by error analysis in Appendix E).

Does the ensemble approach improve the result? The result clarifies that ensemble approaches significantly improve performance. The XLM-R+BanglaBERT-II ensemble achieves an overall μ-F1 of 72.52%, surpassing the single BanglaBERT-II model's μ-F1 of 70.49% by 2.88% and improving Hate Type μ-F1 by 2.33%. The proposed ensemble further enhances performance, achieving an overall μ-F1 score of 73.78% (+1.74% compared to XLM-R+BanglaBERT-II). In task-wise, **BanTriX** excels with Hate Type μ-F1 of 73.38% (+4.87% than XLM-R), Hate Sever-

[2] https://github.com/RJ-Hossan/BLP_T1_2025

Approaches	**Overall**				**Hate Type**		**Hate Severity**		**Targeted Group**	
	Pr(%)	**Re(%)**	**μ-F1(%)**	**TPR(%)**	**μ-F1(%)**	**BER(%)**	**μ-F1(%)**	**BER(%)**	**μ-F1(%)**	**BER(%)**
CNN	64.63	67.77	67.77	41.13	65.39	66.70	71.71	44.68	66.21	65.23
BiLSTM	62.45	66.74	66.74	42.31	64.75	62.82	71.06	45.31	64.40	64.93
BiLSTM+CNN	62.62	64.77	64.77	38.29	61.47	69.17	69.45	46.27	63.39	69.69
BiLSTM+BiGRU	63.95	68.05	68.05	42.93	66.12	62.29	72.00	46.18	66.04	62.75
CNN+BiGRU	67.06	69.50	69.50	47.31	68.30	56.06	72.31	44.30	67.87	57.71
BanglaBERT-I	67.21	67.24	67.24	51.75	65.96	50.50	70.12	43.54	65.63	50.71
BanglaBERT-II	70.70	70.49	70.49	56.52	69.66	47.41	72.54	38.82	69.28	44.22
XLM-R	69.27	70.45	70.45	54.76	69.97	47.65	72.11	40.84	69.27	47.23
mBERT	59.69	66.53	66.53	39.60	63.57	66.02	70.63	50.14	65.40	65.04
(BiLSTM+CNN)+BiLSTM	62.89	69.03	69.03	40.64	67.52	64.11	72.59	48.32	66.97	65.66
BiGRU+(BiLSTM+CNN)	64.79	69.34	69.34	43.33	68.15	62.07	72.59	45.37	67.27	62.57
BanglaBERT-I+XLM-R	69.37	71.49	71.49	51.85	70.84	51.06	73.33	42.14	70.29	51.24
XLM-R+BanglaBERT-II	71.77	72.52	72.52	56.40	71.28	47.35	74.12	39.98	72.17	43.46
Proposed (BanTriX)	71.91	73.78	**73.78**	54.97	**73.38**	47.70	**74.95**	41.07	**73.02**	46.32

Table 2: Performance comparisons on test data across different approaches, where Pr, Re, μ-F1, TPR, and BER denote Precision, Recall, Weighted Micro-F1 score, True Positive Rate, and Balanced Error Rate, respectively.

ity μ-F1 of 74.95%, and Targeted Group μ-F1 of 73.02% (+5.41% than XLM-R), demonstrating that combining Bengali-specific transformers enhances robustness and accuracy across all tasks.

Does CCC loss configuration help? Ablation study in Appendix C shows that the CCC setup (C=Cross-Entropy Loss across three models) achieves the best overall performance, surpassing other loss variants. Poor-performing variants, such as FCW and WFL, indicate heavy overfitting. In a task-wise comparison, CCC provides a more balanced performance across all tasks, demonstrating the effectiveness of the proposed CCC loss function combination in stabilizing training and improving generalization.

6 Conclusion

The study introduced the **BanTriX** architecture tailored for the Bengali multi-task hate speech identification, achieving an overall weighted micro F1-Score of 73.78%. An ablation study highlighted that the optimal configuration, using a token length of 128 with the cross-entropy loss combination, excelled in terms of LIME-based interpretability, confirming its focus on hate-indicative features. Future work will explore the integration of LLMs and advanced techniques, such as dynamic loss optimization, to further enhance rare class detection across diverse datasets.

Limitations

While the study yields strong results for detecting Bengali hate speech, it also presents some apparent limitations. Using a fixed token length of 128 means longer posts may lose important context. The CCC loss setup works well overall; however, it struggles with rare hate categories. It also tends to perform better on predicting *None* cases, which risks missing more subtle hate expressions. Moreover, the study acknowledges class imbalance but does not address it, potentially affecting underrepresented categories. It emphasizes model integration and empirical analysis, relying on pretrained transformers with limited task-specific adaptation or efficiency optimization. Finally, more advanced methods, such as large language models, have yet to be explored.

Ethics Statement

We acknowledge the dataset's annotation biases, though mitigated by clear guidelines and schemas. As it contains only non-identifiable comments, no privacy risks arise. Our model adds no ethical concerns, and fairness was ensured by evaluating across hate types and communities. Overall, the dataset enables hate speech detection to support healthier online discourse, with human oversight mitigating misclassification risks and promoting equitable outcomes.

Acknowledgments

This work is supported by the Directorate of Research and Extension and NLP LAB, Chittagong University of Engineering & Technology (CUET), Chittagong, Bangladesh.

References

Darwin Acharya, Sundeep Dawadi, Shivram Saud, and Sunil Regmi. 2025. Paramananda@NLU of Devanagari script languages 2025: Detection of language, hate speech and targets using FastText and BERT. In *Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL 2025)*, pages 334–338, Abu Dhabi, UAE. International Committee on Computational Linguistics.

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.

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.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in Bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):512–515.

Flor Miriam Plaza del Arco, María Dolores Molina-González, L. Alfonso Ureña-López, and María Teresa Martín-Valdivia. 2021. Comparing pre-trained language models for spanish hate speech detection. *Expert Syst. Appl.*, 166:114120.

Salman Farsi, Asrarul Eusha, Jawad Hossain, Shawly Ahsan, Avishek Das, and Mohammed Moshiul Hoque. 2024. CUET_Binary_Hackers@DravidianLangTech EACL2024: Hate and offensive language detection in Telugu code-mixed text using sentence similarity BERT. In *Proceedings of the Fourth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 193–199, St. Julian's, Malta. Association for Computational Linguistics.

Azizul Hakim Fayaz, MD. Shorif Uddin, Rayhan Uddin Bhuiyan, Zakia Sultana, Md. Samiul Islam, Bidyarthi Paul, Tashreef Muhammad, and Shahriar Manzoor. 2025. Bidwesh: A bangla regional based hate speech detection dataset. *Preprint*, arXiv:2507.16183.

Antigoni Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. *Proceedings of the International AAAI Conference on Web and Social Media*, 12(1).

Jyothish Lal G, Premjith B, Bharathi Raja Chakravarthi, Saranya Rajiakodi, Bharathi B, Rajeswari Natarajan, and Ratnavel Rajalakshmi. 2025. Overview of the shared task on multimodal hate speech detection in Dravidian languages: DravidianLangTech@NAACL 2025. In *Proceedings of the Fifth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 114–122, Acoma, The Albuquerque Convention Center, Albuquerque, New Mexico. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *Preprint*, arXiv:2510.01995.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Ming Shan Hee, Wen-Haw Chong, and Roy Ka-Wei Lee. 2023. Decoding the underlying meaning of multimodal hateful memes. *Preprint*, arXiv:2305.17678.

Eftekhar Hossain, Omar Sharif, and Mohammed Moshiul Hoque. 2022. MUTE: A multimodal dataset for detecting hateful memes. 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: Student Research Workshop*, pages 32–39, Online. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *Preprint*, arXiv:1508.01991.

Vikas Kumar Jha, Hrudya P, Vinu P N, Vishnu Vijayan, and Prabaharan P. 2020. Dhot-repository and classification of offensive tweets in the hindi language. *Procedia Computer Science*, 171:2324–2333. Third International Conference on Computing and Network Communications (CoCoNet'19).

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Preprint*, arXiv:1408.5882.

Thomas Mandl, Koyel Ghosh, Nishat Raihan, Sandip Modha, Shrey Satapara, Tanishka Gaur, Yaashu Dave, Marcos Zampieri, and Sylvia Jaki. 2025. Overview of the hasoc track 2024: Hate-speech identification in english and bengali. In *Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '24, page 12, New York, NY, USA. Association for Computing Machinery.

Abdullah Maruf, Ahmad Jainul Abidin, Md Haque, Zakaria Masud, Aditi Golder, Raaid Alubady, and Zeyar Aung. 2024. Hate speech detection in the bengali language: a comprehensive survey. *Journal of Big Data*, 11.

Arpan Nandi, Kamal Sarkar, Arjun Mallick, and Arkadeep De. 2024. A survey of hate speech detection in indian languages. *Social Network Analysis and Mining*, 14(1):70.

Ahmed Omar, Tarek M. Mahmoud, and Tarek Abd-El-Hafeez. 2020. Comparative performance of machine learning and deep learning algorithms for arabic hate speech detection in osns. In *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*, pages 247–257, Cham. Springer International Publishing.

Nasif Istiak Remon, Nafisa Hasan Tuli, and Ranit Debnath Akash. 2022. Bengali hate speech detection in public facebook pages. In *2022 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 169–173.

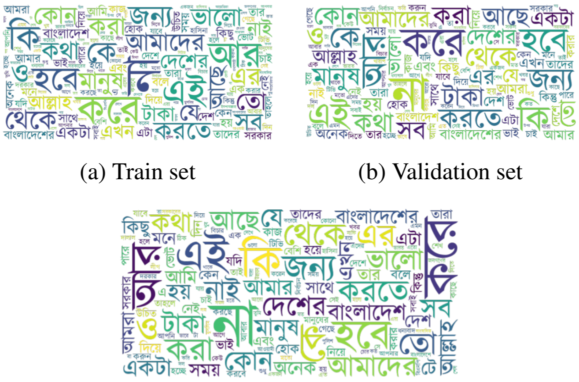
Nauros Romim, Mosahed Ahmed, Hriteshwar Talukder, and Md Saiful Islam. 2020. Hate speech detection in the bengali language: A dataset and its baseline evaluation. *Preprint*, arXiv:2012.09686.

Kengatharaiyer Sarveswaran, Surendrabikram Thapa, Sana Shams, Ashwini Vaidya, and Bal Krishna Bal. 2025. A brief overview of the first workshop on challenges in processing South Asian languages (CHiPSAL). In *Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL 2025)*, pages 1–8, Abu Dhabi, UAE. International Committee on Computational Linguistics.

Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2022. M-BAD: A multilabel dataset for detecting aggressive texts and their targets. In *Proceedings of the Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situations*, pages 75–85, Dublin, Ireland. Association for Computational Linguistics.

A Exploratory Data Analysis

Figure A.1 shows the word cloud generated from the train, validation, and test datasets, where frequently occurring words appear larger in size. This visualization highlights dominant patterns and recurring terms in the corpus, offering quick insights into the lexical distribution of the dataset and serving as an effective tool for understanding key themes and guiding preprocessing decisions.



(a) Train set

(b) Validation set

(c) Test set

Figure A.1: Word clouds (top 200 words) across all three datasets.

Figure A.2 illustrates the feature correlation map, which reveals the three subtasks: Hate Type, Hate Severity, and Targeted Group. They are largely independent, with only weak associations. For example, Hate Type vs. Hate Severity shows a very low positive correlation of 0.07, Hate Type vs. Targeted Group is slightly higher at 0.12, while Hate Severity vs. Targeted Group exhibits a weak

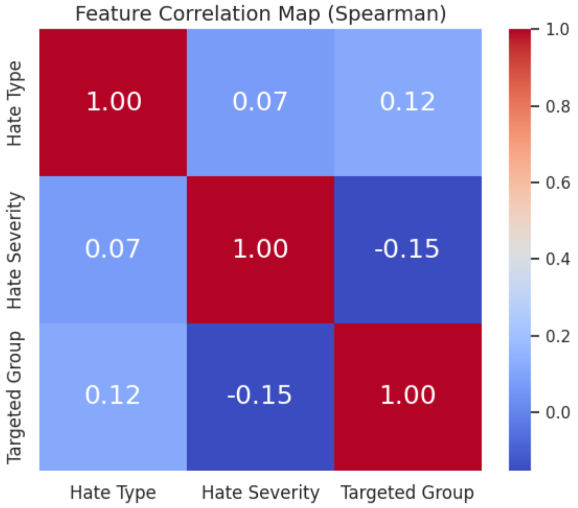


Figure A.2: Feature correlation map across three datasets.

negative correlation of -0.15. These low values indicate that each label contributes distinct information, justifying the multi-task setup and highlighting the datasets richness for modeling diverse aspects of hate speech.

B Tuned Hyperparameters

Table B.1 summarizes the key hyperparameters used for training the proposed architecture (**BanTriX**), including a dropout rate of 0.1, a token length of 128, a batch size of 32, model-specific learning rates (2.5e-5 for BanglaBERT-I, 2e-5 for XLM-R and BanglaBERT-II), AdamW optimizer, linear warmup scheduler with 0 warmup steps, and two training epochs.

Attribute	**Value**
Dropout	0.1
Token Length	128
Batch Size	32
Learning Rate	2.5e-5 (BanglaBERT-I) 2e-5 (XLM-R) 2e-5 (BanglaBERT-II)
Optimizer	AdamW
Scheduler	Linear Schedule with Warmup
Warmup Steps	0
Epochs	2
Loss Function	Cross-Entropy Loss

Table B.1: Hyperparameters used for training of **BanTriX** in multi-task hate speech detection.

C Ablation Study

The ablation study (Table C.1) examines the impact of various loss function combinations and maximum token lengths (T_L) on performance, using an epoch size of 2. The study reveals that the optimal configuration ($T_L = 128$) achieves an overall μ-F1 score of 73.78% and a TPR of 54.97% (slightly lower than the best), thereby balancing context capture and generalization.

C.1 Loss Function Combinations

The proposed architecture uses Cross-Entropy Loss (C) across three transformer models (BanglaBERT-I, XLM-R, BanglaBERT-II). Still, we explored combinations of Cross-Entropy (C), Focal Loss (F), Weighted Cross-Entropy (W), and Label-Smoothed Cross-Entropy (L), as defined in Eq. C.7.

$$\mathcal{L}_{CE} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \tag{C.7a}$$

$$\mathcal{L}_{WCE} = -\sum_{c=1}^{C} w_c y_c \log(\hat{y}_c), \tag{C.7b}$$

$$\mathcal{L}_{FL} = -\sum_{c=1}^{C} (1-\hat{y}_c)^{\gamma} y_c \log(\hat{y}_c) \tag{C.7c}$$

$$\begin{aligned}\mathcal{L}_{LSCE} = -(1-\epsilon)\sum_{c=1}^{C} y_c \log(\hat{y}_c) \\ - \tfrac{\epsilon}{C}\sum_{c=1}^{C} \log(\hat{y}_c)\end{aligned} \tag{C.7d}$$

where reduction="mean", $\gamma = 2.0$, $\alpha =$ None, and $\epsilon = 0.1$ used.

The **CFL** setup (Cross-Entropy, Focal Loss, Label-Smoothed) achieves the best results with 73.03% μ-F1 and 53.64% TPR, surpassing **FFF** (all Focal Loss) at 72.52% μ-F1 and 55.23% TPR by +0.70% μ-F1 and -2.96% TPR. Poor-performing variants **FCW**, **WFL**, and **LWF** yield only 32.05% μ-F1 and 23.33% TPR, reflecting heavy overfitting. In task-wise, CFL records 72.40% μ-F1 for Hate Type (vs. 71.57% of FFF), 74.68% for Severity (vs. 74.46%), and 72.00% for Targeted Group (vs. 71.53%), with slightly higher BER (+5.87%) on Hate Type but overall more balanced performance after the proposed `CCC` loss function combination.

C.2 Maximum Token Length

Varying the maximum token length (T_L) impacts context capture. At T_L of 156, the proposed method achieves the 2$^{\text{nd}}$ best (after T_L of 128) overall μ-F1 of 73.72% and TPR of 56.72%, surpassing T_L of 64 (73.51% μ-F1) by 0.29% in μ-F1, and T_L of 256 (72.91% μ-F1, 59.16% TPR) by +1.11% in μ-F1 but -4.30% in TPR. In task-wise, T_L of 156 yields Hate Type μ-F1 of 72.94% (0.27% better than T_L of 64) with BER of 47.58%, Hate Severity μ-F1 of 75.21% with BER of 38.53%, and Targeted Group μ-F1 of 73.01%, indicating optimal performance (except token length of 128) at token length of 156.

C.3 Batch Size Analysis

Figure C.1 shows that a batch size of 32 yields the best results, with μ-F1 scores of 73.38% (Hate

Attributes	Overall				Hate Type		Hate Severity		Targeted Group	
	Pr(%)	Re(%)	μ-F1(%)	TPR(%)	μ-F1(%)	BER(%)	μ-F1(%)	BER(%)	μ-F1(%)	BER(%)
Loss Function Combinations										
CFL	71.01	73.03	73.03	53.64	72.40	49.39	74.68	40.69	72.00	48.99
FCW	16.44	32.05	32.05	23.33	22.67	83.33	66.05	66.67	7.44	80.00
WFL	16.44	32.05	32.05	23.33	22.67	83.33	66.05	66.67	7.44	80.00
FFF	71.35	72.52	72.52	55.23	71.57	46.65	74.46	40.08	71.53	47.58
LWF	16.44	32.05	32.05	23.33	22.67	83.33	66.05	66.67	7.44	80.00
Maximum Token Length, T_L										
T_L **= 64**	72.40	73.51	73.51	57.32	72.74	46.41	74.87	39.26	72.91	42.37
T_L **= 128**	71.91	73.78	73.78	54.97	73.38	47.70	74.95	41.07	73.02	46.32
T_L **= 156**	72.71	73.72	73.72	56.72	72.94	47.58	75.21	38.53	73.01	43.74
T_L **= 224**	71.89	73.57	73.57	55.17	72.81	47.84	75.20	40.17	72.70	46.48
T_L **= 256**	73.19	72.91	72.91	59.16	72.08	45.14	74.28	36.80	72.37	40.56

Table C.1: Ablation study of **BanTriX** on test data with loss function combinations and maximum token length. Here, Pr, Re, μ-F1, TPR, and BER denote Precision, Recall, Weighted Micro-F1 score, True Positive Rate, and Balanced Error Rate, respectively.

Type), 74.95% (Severity), and 73.02% (Targeted Group), along with the lowest BERs. Smaller sizes (8, 16) achieve around 72–74.6% μ-F1, while larger sizes (64, 96) drop to about 69–74.1% μ-F1 with higher BERs, confirming batch size 32 as the optimal choice.

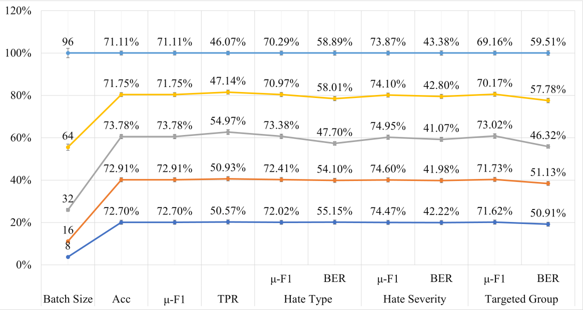


Figure C.1: Overview of the batch size impact while the maximum token length is 128 and cross-entropy loss is used.

D Performance Comparison

Table D.1 compares teams' performance with baselines in the task, with *CUET-NLP_Zenith* (our team) achieving a Weighted Micro F1-Score (μ-F1) of 73.78%, ranking second. It trails *mahim_ju's* performance by 0.16% but beats *shifat_islam's* performance by 0.23%. Compared to the baselines provided by the organizers, the proposed **BanTriX** outperforms the Majority Baseline by 21.51% and the n-gram Baseline by 17.02%, demonstrating its superiority over traditional approaches.

Baseline/Team	**μ-F1 (%)**	**Rank**
mahim_ju	73.92	1
CUET-NLP_Zenith	73.78	2
shifat_islam	73.61	3
reyazul	73.32	4
Random Baseline	23.04	-
Majority Baseline	60.72	-
n-gram Baseline	63.05	-

Table D.1: Performance comparison of the proposed architecture with other teams' approaches.

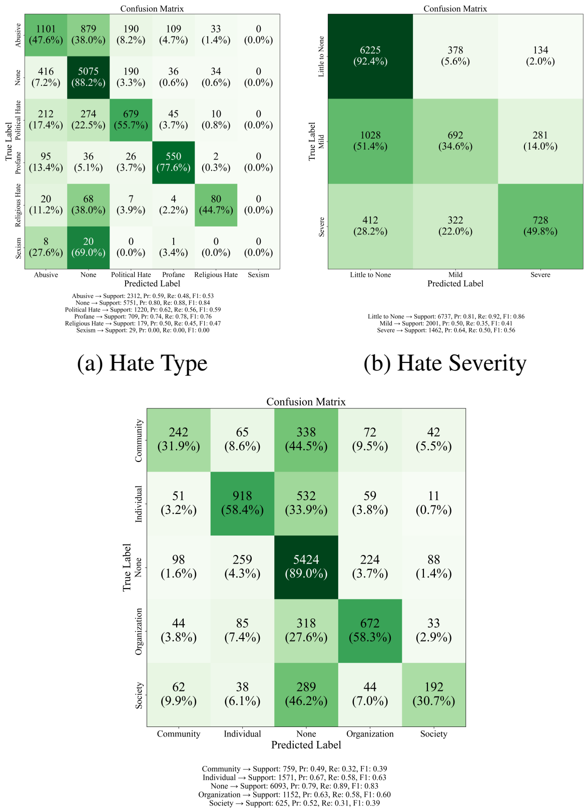
E Error Analysis

A thorough quantitative and qualitative error analysis was conducted to gain an in-depth understanding of the proposed architecture's performance in the task.

E.1 Quantitative Analysis

The confusion matrices presented in Figure E.1 reveal distinct performance patterns across hate types (Figure E.1a), hate severity (Figure E.1b), and targeted groups (Figure E.1c).

For *hate severity*, the model performs well on *Little to None* (92.4% accuracy, F1 of 0.86) but struggles with *Mild* (F1 of 0.41, often misclassified as *Little to None*) and *Severe* (F1 of 0.56, with frequent downgrading). For targeted groups, performance is weak due to dataset skew, with *Society* achieving only 0.39 F1 and few instances for *Community*. Regarding hate types, strong results are observed for *Profane* (F1 of 0.76) and *None* (F1 of 0.84). In contrast, *Abusive* exhibits limited recall (0.48), and *Sexism* performs poorly, often



(a) Hate Type

(b) Hate Severity

(c) Targeted Group

Figure E.1: Confusion matrices for different categories in the task.

being misclassified as *None*. Overall, the model excels at the majority classes but struggles with minority hate categories (e.g., *sexism*), reflecting bias toward predicting *None* in ambiguous cases.

E.2 Qualitative Analysis

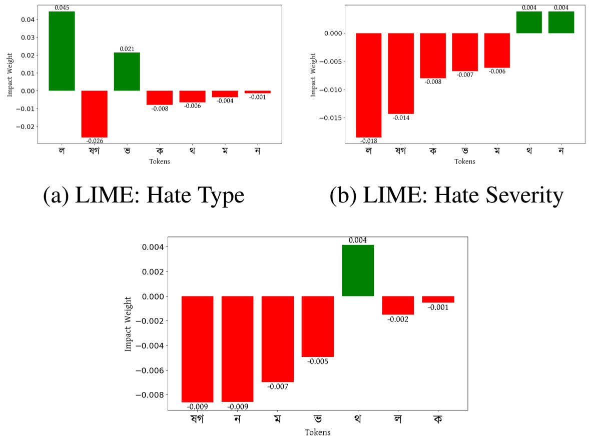
The qualitative analysis of sample classifications shown in Table E.1 illustrates varied model performance in detecting hate speech in Bengali YouTube comments. The model demonstrates strong performance on neutral samples (IDs 266764 and 653626), both labeled as *None*, where the predictions match perfectly. This indicates robustness in handling straightforward non-hate content. In contrast, it struggles with nuanced cases, e.g., sample 241030 (*Political Hate*) was misclassified as *Abusive*, likely due to overlapping sarcastic or abusive tones, while sample 742298 (*Abusive*) was predicted as *None*, reflecting difficulties with cultural subtleties and dataset imbalance. Overall, the model reliably detects clear non-hate instances but faces challenges with context-dependent hate and minority categories.

Sample No.	Text	#	Hate Type	Hate Severity	Targeted Group
241030	ভারতীয় দালাল সময় টিভিকে বয়কট করুন	Actual	*Political Hate*	*Mild*	*Organization*
		Prediction	***Abusive*** ✗	✓	✓
266764	অস্ত্র সহ সেনাবাহিনী পাঠানো হোক	Actual	*None*	*Little to None*	*None*
		Prediction	✓	✓	✓
742298	আর্জেন্টিনা ১৯৮৬ আর ১৯৯০ এ কি করছে আবুল সাংঘাতিক	Actual	*Abusive*	*Mild*	*Individual*
		Prediction	***None*** ✗	***Little to None*** ✗	***None*** ✗
653626	ভালো মানুষগুলো ভালো থাকুক	Actual	*None*	*Little to None*	*None*
		Prediction	✓	✓	✓

Table E.1: Few sample predictions by **BanTriX** in the task. The $\surd$ mark indicates the correct predictions, and *X* denotes incorrect predictions.

F Model Interpretability

The LIME-based explanation bar plots (see Figure F.1) for the last sample of Table E.1, labeled as *None* across hate type, severity, and targeted group, provide insights into token contributions to the model's predictions. For hate severity, the plot shows that tokens like "ল *(la)*" and "ক *(ka)*" negatively impact (red bars) the prediction of *Little to None* severity, reducing its likelihood, while other tokens positively impact (green bars), supporting the *None* prediction. In the hate type plot, "ন *(na)*" negatively affects the predictions, whereas some tokens like "ল *(la)*", "ভ *(bha)*" positively reinforce the *None* classification, highlighting these tokens as key neutral indicators. Overall, the model relies heavily on neutral tokens to correctly classify this sample as *None* across all categories.



(a) LIME: Hate Type

(b) LIME: Hate Severity

(c) LIME: Targeted Group

Figure F.1: LIME-based model interpretability for the last sample of Table E.1.

TeamHateMate at BLP Task1: Divide and Conquer: A Two-Stage Cascaded Framework with K-Fold Ensembling for Multi-Label Bangla Hate Speech Classification

Mehedi Hasan, Mahbub Islam Mahim
Jahangirnagar University, Savar, Dhaka
These authors contributed equally to this work.

Abstract

Detecting hate speech on social media is essential for safeguarding online communities, yet it remains challenging for low-resource languages like Bangla due to class imbalance and subjective annotations. We introduce a two-stage cascaded framework with k-fold ensembling to address the BLP Workshop 2025 Shared Task's three subtasks: 1A (hate type classification), 1B (target identification), and 1C (joint classification of type, target, and severity). Our solution balances precision and recall, achieving micro-$F1$ scores of 0.7331 on 1A, 0.7356 on 1B, and 0.7392 on 1C, ranking 4th on 1A and **1st** on both 1B and 1C. It performs strongly on major classes, although underrepresented labels such as *sexism* and *mild severity* remain challenging. Our method makes the optimal use of limited data through k-fold ensembling and delivers overall balanced performance across majority and minority classes by mitigating class imbalance via cascaded layers.

1 Introduction

Social media's growth has accelerated the spread of hate speech, presenting major threats to online safety and public welfare (Vogels, 2021). Despite having a large speaker base, Bangla is still not well-studied, even though automatic detection systems have advanced in high-resource languages (Fortuna and Nunes, 2018; Das et al., 2021). Subjectivity in annotation, overlapping linguistic cues across categories, and a stark class imbalance make the task especially challenging (Vidgen and Derczynski, 2020).

To overcome these obstacles, we introduce a two-phase cascaded framework with k-fold ensembling in this paper (Tang and Dalzell, 2019). After distinguishing between hate and non-hate content, our method gradually improves predictions across multiple categories. Our approach reduces overfitting and stabilizes performance in imbalanced conditions by using ensembling and cross-validation (Mozafari et al., 2020). Our framework shows strong results across all subtasks of the BLP shared task 1 (Hasan et al., 2025b), demonstrating that our divide-and-conquer tactics can successfully maintain a balance between recall and precision for Bangla hate speech detection.

2 Related Work

Recent studies approach hate speech detection in low-resource languages using transformer models, hybrid architectures, and ensemble techniques. (Saha, 2023) combined IndicBERT with a Naive Bayes classifier and synthetic upsampling, achieving macro-F1 scores of 0.73 (Assamese), 0.68 (Bengali), and 0.84 (Bodo). Their approach of data augmentation using back translation tended to smooth out language-specific features, reducing multi-class separability in our study. (Ripoll et al., 2022) used multilingual transformer models trained across k-fold splits and ensembled via soft voting. Their system ranked first place in contextual hate speech, but lacks addressing class imbalance issues. (Das et al., 2023) proposed a Hierarchical-BERT for Bangla violence detection, where the first layer identifies major classes and the second layer refines them. However, misclassifications in the first layer often propagate errors, whereas our cascading design allows the second layer to revise initial mispredictions. (Veeramani et al., 2023) ensembled three BERT-based models with distinct optimization strategies such as extra classification heads and masked language model pretraining. Their system achieved F1 scores of 0.7347 for violence and 0.7173 for sentiment. (Ababu et al., 2025) applied BiLSTM with FastText embeddings for bilingual hate speech detection in Amharic and Afaan Oromo, reaching 78.05% accuracy. While FastText helped them deal with words that were out of their vocabulary, its static nature hindered contextual understanding—a crucial component of multi-class

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 453–460
December 23, 2025 ©2025 Association for Computational Linguistics

hate classification.

3 Task Description

3.1 Task Overview

This shared task focuses on detecting hate speech in Bangla social media content across three subtasks: 1A - classify the type of hate (e.g., Abusive, Sexism, Religious, Political, Profane, or None); 1B - identify the target (Individuals, Organizations, Communities, or Society); 1C - jointly predict hate type, severity (little to none, mild, severe), and target in a multi-task setup. Subtasks 1A and 1B are evaluated using the Micro-F1 score to address class imbalance. Subtask 1C is evaluated using the average Micro-F1 across the hate type, severity, and target predictions.

3.2 Dataset Overview

We used the workshop's (Hasan et al., 2025a) YouTube comments dataset (35.5k rows), which mirrors real-world trends where non-hate content clearly dominates. Specifically, 56.28% of comments fall under "None" for `hate_type`, 66.24% are rated "Little to None" for `severity`, and 59.75% target no particular group. In contrast, the rarest labels are *Sexism* at just 0.35%, *Severe* at 14.48%, and *Society* at 6.17%.

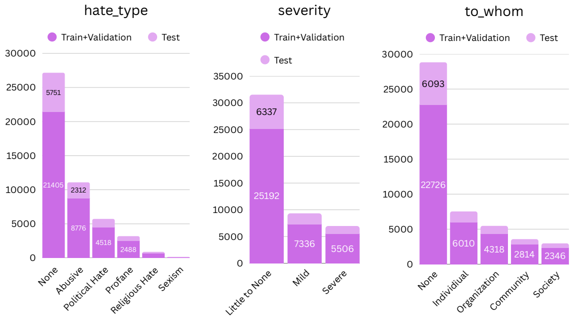


Figure 1: Distribution of classes

4 Methodology

We used a **cascading framework** to boost confidence and reduce error rates in major classes, raising the overall F1 score. Each subtask runs in two stages. Stage 1 separates hate from non-hate content with a binary classifier. Stage 2 applies multi-class classification to the samples labeled as hate in Stage 1. This stage can also reclassify content flagged as hate back into non-hate classes, giving any false positives a second chance. This setup helps the model generalize better across all categories. Because each level refines the previous one instead of forming a strict hierarchy, we call it cascading levels (Figure 2).

We selected **evaluation metrics** for each stage's needs. Stage 1's binary classifier uses the F_2-micro score, prioritizing recall to reduce false negatives in non-hate detection. Stage 2's multi-class classifier uses the F_1-micro score to balance precision and recall across all classes and ensure fair, accurate label distribution. These choices align with our cascading framework's goal of initial high-confidence filtering of major classes followed by comprehensive classification of all classes.

We used k**-fold cross-validation with an ensemble approach** (Figure 2) on each stage to improve generalization and make full use of our data. We merged the training and validation sets, then split them into folds. For Subtasks 1A and 1B, we set $k = 7$; for Subtask 1C's severity, $k = 9$. We applied `StratifiedKFold` to keep class ratios balanced across folds, producing k models at each stage. During inference, we ensembled those models by majority vote. In multi-class tasks, ties were broken by choosing the most frequent class in the training set.

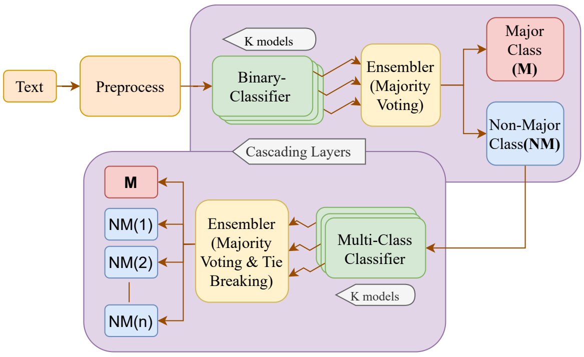


Figure 2: Cascading classification architecture with level-wise k-fold ensembling

To improve classification performance for Subtask 1B, we used an **enhanced `BanglaBERT` architecture** for the 2nd stage of the multi-class classifier (Figure 3). We processed each input by first encoding it into token IDs and attention masks, which were then passed through the pretrained BanglaBERT encoder to obtain contextual embeddings. To summarize token-level information, we applied an attention-pooling mechanism that dynamically weighted important tokens, producing a single pooled vector per instance. This pooled output was regularized using dropout and transformed

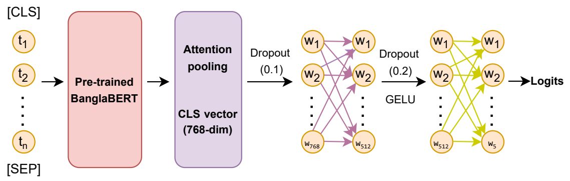


Figure 3: Enhanced BanglaBERT with Attention Pooling and Classification Head

via a dense layer followed by GELU activation to introduce non-linearity. A second dropout layer was applied to further reduce overfitting before final classification. The transformed features were then mapped to class logits using a linear layer. During training, we computed loss using CrossEntropy with optional label smoothing to improve stability on imbalanced data.

For Subtask 1C, we developed **three independent classification pipelines** shown in Figure 4, each trained separately; their outputs were then aggregated to improve the average micro-F1 score.

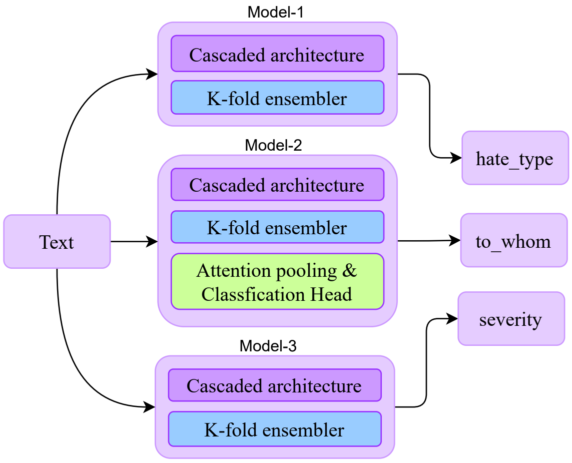


Figure 4: Aggregation of 3 models

5 Results and Findings

We evaluated all tasks using the final test dataset provided by the workshop organizers. Each subtask test set contains **10,200** instances.

The overall performance across the three subtasks is shown in Table 1. With F_1 scores of **0.7331**, **0.7356**, and **0.7392** for Subtasks 1A, 1B, and 1C, our cascaded framework with k-fold ensembling consistently delivers strong results. The reliability of our divide-and-conquer strategy is demonstrated by the improvements, which are not only evident in terms of F_1 but also balanced across precision and recall.

Subtask	Accuracy	Precision	Recall	F1 Score
1A	0.7331	0.7235	0.7331	0.7331
1B	0.7356	0.7276	0.7356	0.7356
1C	0.7392	0.7266	0.7392	0.7392

Table 1: Overall performance metrics across all subtasks.

For hate type classification, our system achieved an overall F_1 of **0.7331** with precision $= 0.7439$ and recall $= 0.7395$. As shown in Table 2, the *None* class achieved the highest performance ($F_1 =$ 0.8361, precision $= 0.8221$, recall $= 0.8506$), indicating the effectiveness of the first-stage filtering in distinguishing non-hate content. Profane language also scored strongly ($F_1 = 0.7500$). However, performance was weaker for minority categories, particularly *Religious Hate* ($F_1 = 0.4531$) and *Sexism* (no correct predictions). The challenges with the *Sexism* category stem from its extreme underrepresentation, with only 9 examples in a training set of over 35k. In the test set, the first-stage classifier detected just 11 of 29 sexism instances (recall $= 0.38$), and none of these were correctly labeled as *Sexism* in the second stage, often falling under broader categories like *Abusive*.

Class	Precision	Recall	F1 Score
Abusive	0.5761	0.5272	0.5506
None	**0.8138**	**0.8597**	**0.8361**
Political Hate	0.6175	0.5730	0.5944
Profane	0.7309	0.7701	0.7500
Religious Hate	0.5385	0.3911	0.4531
Sexism	0.0000	0.0000	0.0000

Table 2: Per-class performance metrics for Subtask 1A.

In hate target prediction, our system obtained an overall F_1 of **0.7356**, with precision $= 0.7510$ and recall $= 0.7394$. Similar to Subtask 1A, the *None* category was detected with high performance ($F_1 = 0.8408$, recall $= 0.8611$), showing the effectiveness of cascading from the binary stage and the influence of the training dataset. Among hate-bearing categories, *Individual* ($F_1 = 0.6446$) and *Organization* ($F_1 = 0.6040$) were captured reasonably well. By contrast, *Community* ($F_1 = 0.4470$) and *Society* ($F_1 = 0.4499$) proved harder to detect, suggesting subtler linguistic markers and limited representation in the dataset.

The third subtask differs from the previous ones as it requires predicting hate type, target, and severity jointly. We reuse outputs from Subtasks 1A

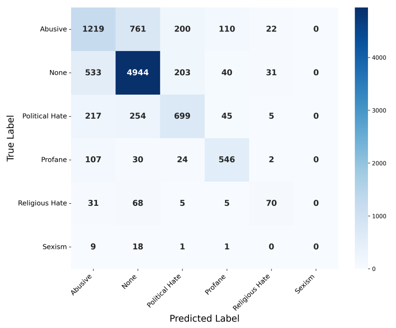


Figure 5: Confusion matrix for Subtask 1A.

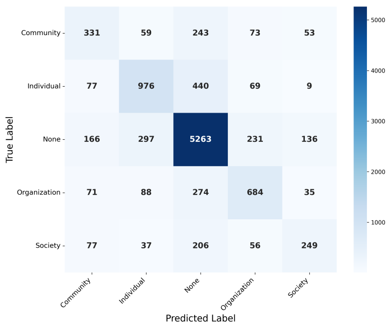


Figure 6: Confusion matrix for Subtask 1B.

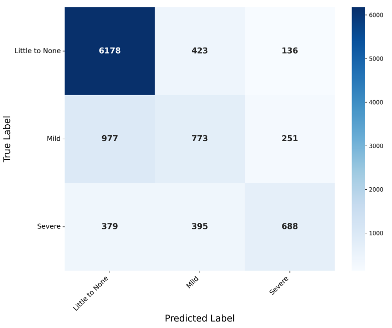


Figure 7: Confusion matrix for Subtask 1C.

Class	Precision	Recall	F1 Score
Community	0.4584	0.4361	0.4470
Individual	0.6699	0.6213	0.6446
None	**0.8190**	**0.8638**	**0.8408**
Organization	0.6146	0.5938	0.6040
Society	0.5166	0.3984	0.4499

Table 3: Per-class performance metrics for Subtask 1B.

and 1B, focus solely on severity here, then merge all three labels, simplifying the workflow and maintaining consistency. Our system achieved an overall F_1 of **0.7392** (precision $= 0.7471$, recall $= 0.7390$). It excelled on Little to None ($\mathrm{F}_1 = 0.8658$, recall $= 0.9170$) but lagged on Severe ($\mathrm{F}_1 = 0.5424$) and Mild ($\mathrm{F}_1 = 0.4304$). This reflects the challenge of distinguishing between mild and severe hate, where subjectivity in annotation and overlapping cues complicate classification.

Class	Precision	Recall	F1 Score
Little to None	**0.8200**	**0.9170**	**0.8658**
Mild	0.4859	0.3863	0.4304
Severe	0.6400	0.4706	0.5424

Table 4: Per-class performance metrics for Subtask 1C - Hate Severity only.

The confusion matrices (Figures 5, 6, and 7) show systematic misclassification trends. In Subtask 1A, many instances of *Abusive*, *Profane*, and *Political Hate* are misclassified to the *None* class, with additional overlap between *Abusive* and both *Political Hate* and *Profane*. Subtask 1B shows a similar pattern, where most categories are misclassified as *None*, alongside confusions among *Community*, *Organization*, and *Society*. In Subtask 1C, errors largely reduce to *Little to None*, but we also observe overlap between *Mild* and *Severe*, with severe cases often predicted as mild. Together, these patterns underline both the dominance of neutral labels and the difficulty of separating closely related categories.

Key findings and overall insights:

- Subtask 1A and 1B results show that the system is highly effective at detecting the *None* category (non-hate content), which boosts global performance, but struggles with minority classes such as *Sexism*, *Religious Hate*, and *Community*; Subtask 1C reuses hate type and target labels, focuses on severity, and achieves an overall F_1 of 0.7392.

- Precision and recall remain high for dominant categories (e.g., *None*, *Little to None*), but recall drops significantly for underrepresented or ambiguous categories such as *Mild* severity, where annotation subjectivity is likely a factor.

- The cascaded framework achieves strong performance across all subtasks with balanced precision and recall; k-fold ensembling optimizes the use of limited low-resource language data to enhance overall results.

6 Conclusion

In this work, we proposed a two-stage cascaded framework for multi-label classification of hate speech in Bangla using k-fold ensembling. Our system ranked among the top submissions (4th, 1st, and 1st) in the shared task, consistently achieving balanced precision, recall, and F1-score across hate type, target, and severity. Although there are still issues with underrepresented and ambiguous labels like "Sexism" and "Mild severity," the framework was especially successful at handling dominant categories and filtering non-hate content. These results illustrate the potential of cascaded ensembling for low-resource hate speech detection as well as the ongoing requirement for more balanced, richer datasets to capture subtle types of online abuse.

Limitations

The system lacks the implementation of contrastive learning. As the proposed problem to be solved has overlapping classes, contrastive learning could help. It pulls similar examples together and pushes dissimilar ones apart. This creates fine-grained distinctions between close classes like community and organizations.

The system lacks proper data augmentation. Minority classes suffer as a result. The model does well on dominant labels but fails to detect labels such as Sexism accurately. Per-class F1 scores drop despite strong global metrics.

Critical Analysis

We experimented with data augmentation through back-translation, but this strategy did not improve the performance of the models, most likely due to overlapping class annotations that limited the value of augmented samples.

We also explored an enhanced classification head for Subtasks 1A and 1C. Although initial experiments involved some hyperparameter tuning, the limited scope of training time and resources prevented us from fully optimizing the approach, and the gains remained inconsistent. Consequently, it was not included in our final system. In contrast, the same design showed clearer improvements in Subtask 1B, suggesting that with more extensive hyperparameter exploration, it could potentially benefit Subtasks 1A and 1C as well.

Lastly, model generalization was limited by dataset-specific issues like class imbalance and inconsistent annotations. Although methods such as weighted class and over-sampling were tried, they were unable to completely counteract the test dataset's bias toward majority classes.

References

Teshome Mulugeta Ababu, Michael Melese Woldeyohannis, and Emuye Bawoke Getaneh. 2025. Bilingual hate speech detection on social media: Amharic and afaan oromo. *Journal of Big Data*, 12(1):30.

Amit Kumar Das, Abdullah Al Asif, Anik Paul, and Md Nur Hossain. 2021. Bangla hate speech detection on social media using attention-based recurrent neural network. *Journal of Intelligent Systems*, 30(1):578–591.

Udoy Das, Karnis Fatema, Md Ayon Mia, Mahshar Yahan, Md Sajidul Mowla, Md Fayez Ullah, Arpita Sarker, and Hasan Murad. 2023. Emptymind at blp-2023 task 1: A transformer-based hierarchical-bert model for bangla violence-inciting text detection. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 174–178, Singapore. ACL.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *Acm Computing Surveys (Csur)*, 51(4):1–30.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. 2020. Hate speech detection and racial bias mitigation in social media based on bert model. *PloS one*, 15(8):e0237861.

Maria Luisa Ripoll, Fadi Hassan, Joseph Attieh, Guillem Collell, and Abdessalam Bouchekif. 2022. Multi-lingual contextual hate speech detection using transformer-based ensembles. In *Proceedings of the Fourth Workshop on Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC 2022)*, volume 3395 of *CEUR Workshop Proceedings*. CEUR-WS.org.

et al. Saha. 2023. Ensemble indicbert with naive bayes classifier and synthetic upsampling for hate-speech detection. In *Proceedings of CEUR Workshop*.

Yiwen Tang and Nicole Dalzell. 2019. Classifying hate speech using a two-layer model. *Statistics and Public Policy*, 6(1):80–86.

Hariram Veeramani, Surendrabikram Thapa, and Usman Naseem. 2023. Lowresourcenlu at blp-2023 task 1 & 2: Enhancing sentiment classification and violence incitement detection in bangla through aggregated language models. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 230–235, Singapore. ACL.

Bertie Vidgen and Leon Derczynski. 2020. Directions in abusive language training data, a systematic review: Garbage in, garbage out. *Plos one*, 15(12):e0243300.

Emily A Vogels. 2021. *The state of online harassment*, volume 13. Pew Research Center Washington, DC.

A Source Code

To foster reproducibility and future research, we release all implementation source code and resources at `https://github.com/mahbubislammahim/Bangla-Hate-Speech-Identification`

B Experimental Setup

B.1 Hardware and Runtime

All training experiments were performed using an NVIDIA GeForce RTX 4090 GPU with 24 GB memory. Additional experiments were occasionally run on free GPUs: Google Colab (T4, 16 GB) and Kaggle (P100 or T4, 15 GB). Training times were around an hour.

B.2 Text Normalization and Tokenization

All inputs are normalized using the BUET NLP normalizer to reduce script-level noise and standardize punctuation/spacing. Tokenization is performed with the `csebuetnlp/banglabert` WordPiece tokenizer, with sequences padded or truncated to a maximum of 256 tokens.

B.3 Cascaded Modeling

Each subtask uses a two-stage cascaded framework: stage 1 routes an input either to the "none" category or forwards it to Stage 2 for fine-grained classification. Stage 2 predicts the task-specific labels.

For Subtask 1A and 1C, both stages employ the BanglaBERT base model with standard configuration. For Subtask 1B, Stage 2 uses an enhanced classification head with attention pooling and a dense 512-GELU layer.

B.4 Training Protocol

We used the HuggingFace `Trainer` with AdamW, linear learning-rate scheduling with warmup, gradient accumulation, mixed precision (FP16), and weight decay. Stage 1 optimizes micro-F_2, while Stage 2 optimizes micro-F1. Cross-validation is stratified (`random_state=42`) and shuffling is enabled. Stage 1 models are selected by the best validation micro-F_2. Stage 2 models are selected by the best validation micro-F_1.

B.5 Inference and Ensemble Strategy

- **Stage 1:** average positive-class probability across folds.
- **Routing:** threshold τ decides between "none" vs. forwarding to Stage 2 ($\tau = 0.5$ for 1A/1B, $\tau = 0.3$ for 1C).
- **Stage 2:** per-fold argmax predictions are aggregated via majority vote; ties are broken deterministically using task-specific priority orders.

B.6 Hyperparameter Tuning

We adopt a pragmatic tuning strategy:

1. Base learning rate: 3×10^{-5} for Stage 1; 3–4$\times 10^{-5}$ for Stage 2.
2. Short training (2–3 epochs) with gradient accumulation.
3. Label smoothing of 0.1 for Subtasks 1A/1B to stabilize predictions and break ties.
4. Step-based validation for volatile multi-class heads.

B.7 Subtask Configurations

Subtask 1A (Hate Type; 6 classes) 7-fold CV (approx. 5074 rows/fold from 35522 total); Stage 1: {None, Hate}; Stage 2: {None, Religious Hate, Sexism, Political Hate, Profane, Abusive}.

Subtask 1B (Target; 5 classes) 7-fold CV (approx. 5074 rows/fold from 35522 total); Stage 1: {None, Hate}; Stage 2: {None, Society, Organization, Community, Individual}. Stage 2 uses enhanced head with attention pooling.

Subtask 1C (Severity; 3 classes) 9-fold CV (approx. 3946 rows/fold from 35522 total); Stage 1: {Little-to-None, Has-Severity}; Stage 2: {Little-to-None, Mild, Severe}. Stage 2 evaluates every 500 steps with $\tau = 0.3$ for improved recall. In this subtask, final outputs are merged into a single submission by joining 1A (hate type), 1B (target), and severity predictions on `id`.

B.8 Environment and Reproducibility

We use `transformers` ($\geq$ 4.21), `datasets` ($\geq$ 2.0), `accelerate` ($\geq$ 0.20), `PyTorch` ($\geq$ 1.12), `scikit-learn` (0.24), and `numpy` (1.20). Seeds are fixed at 42.

B.9 Training Analysis

To better illustrate the training process, we present example diagrams for Subtask 1B in Figure 8. We also include attention heatmap visualizations for 1B and 1C in Figure 9 and Figure 10, respectively. This demonstrates how the model attends to input tokens and provide insight into its decision-making process.

Hyperparameter	**Stage 1 (Binary)**	**Stage 2 (6-way)**
CV folds	7	7
Base model	csebuetnlp/banglabert	csebuetnlp/banglabert
Max seq. length	256	256
Optimizer / Scheduler	AdamW / Linear	AdamW / Linear
Learning rate	3×10^{-5}	3×10^{-5}
Epochs	2	2
Batch size (train / eval)	16 / 16	8 / 8
Grad. accumulation	2	2
Warmup ratio	0.1	0.1
Weight decay	0.01	0.01
Label smoothing	0.1	0.1
FP16	True	True
Ensemble	Prob. avg.	Majority vote + tie-break
Routing threshold τ	0.5	–

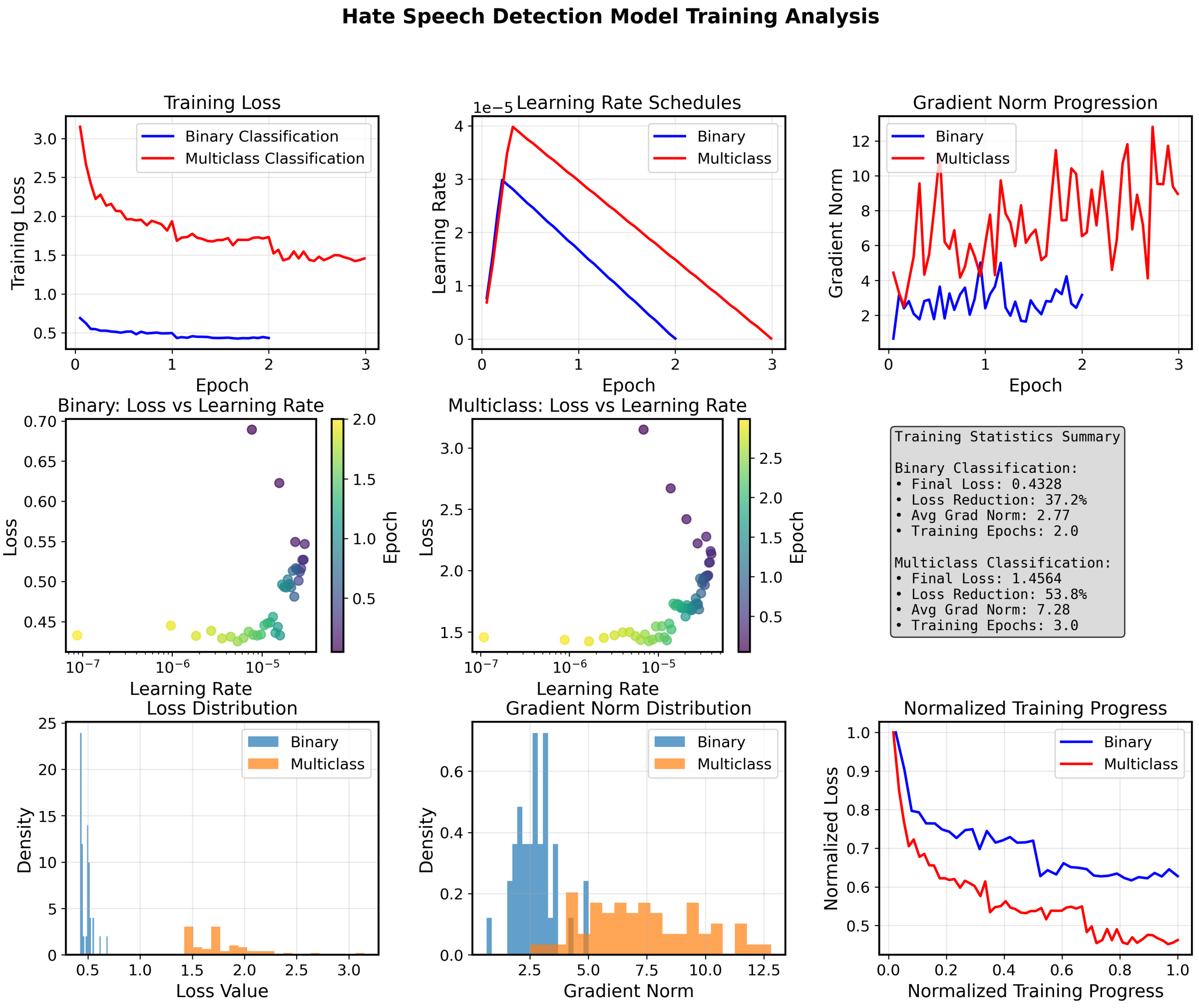
Table 5: Subtask 1A: Training setup.

Hyperparameter	**Stage 1 (Binary)**	**Stage 2 (5-way, Enhanced)**
CV folds	7	7
Base model	csebuetnlp/banglabert	csebuetnlp/banglabert + attention pooling
Head	Standard	Dense(512)+GELU + Dropout + Linear
Max seq. length	256	256
Optimizer / Scheduler	AdamW / Linear	AdamW / Linear
Learning rate	3×10^{-5}	4×10^{-5}
Epochs	2	3
Batch size (train / eval)	16 / 16	16 / 16
Grad. accumulation	2	2
Warmup ratio	0.1	0.1
Weight decay	0.01	0.01
Label smoothing	0.1	0.1
FP16	True	True
Ensemble	Prob. avg.	Majority vote + tie-break
Routing threshold τ	0.5	–

Table 6: Subtask 1B: Training setup.

Hyperparameter	**Stage 1 (Binary)**	**Stage 2 (3-way)**
CV folds	9	9
Base model	csebuetnlp/banglabert	csebuetnlp/banglabert
Max seq. length	256	256
Optimizer / Scheduler	AdamW / Linear	AdamW / Linear
Learning rate	3×10^{-5}	4×10^{-5}
Epochs	2	3
Batch size (train / eval)	16 / 16	16 / 16
Grad. accumulation	2	2
Warmup ratio	0.1	0.1
Weight decay	0.01	0.01
Label smoothing	–	–
FP16	True	True
Ensemble	Prob. avg.	Majority vote + tie-break
Routing threshold τ	0.3	–

Table 7: Subtask 1C: Training setup.



Training Statistics Summary

Binary Classification:
- Final Loss: 0.4328
- Loss Reduction: 37.2%
- Avg Grad Norm: 2.77
- Training Epochs: 2.0

Multiclass Classification:
- Final Loss: 1.4564
- Loss Reduction: 53.8%
- Avg Grad Norm: 7.28
- Training Epochs: 3.0

Figure 8: Training analysis.

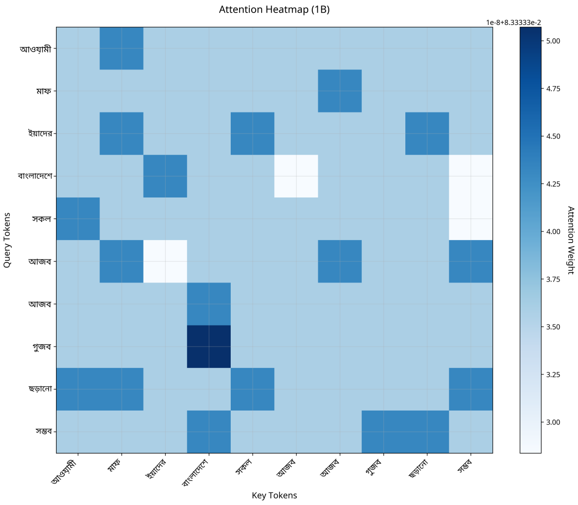


Figure 9: Attention heatmap for Subtask 1B.

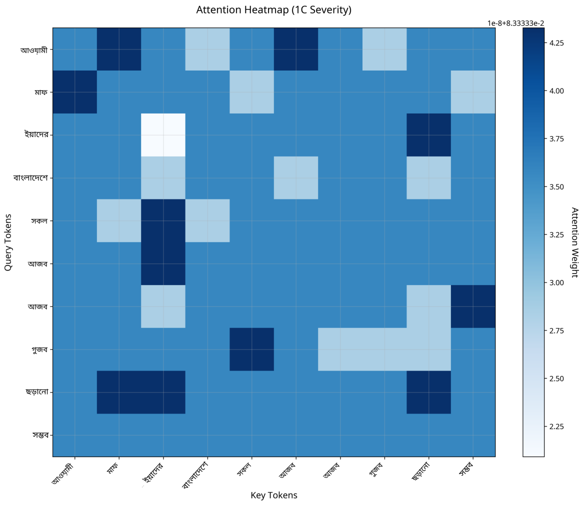


Figure 10: Attention heatmap for Subtask 1C.

NSU_MILab at BLP-2025 Task 1: Decoding Bangla Hate Speech: Fine-Grained Type and Target Detection via Transformer Ensembles

Md. Mohibur Rahman Nabil[1], Muhammad Rafsan Kabir[1], Rakibul Islam[2], Fuad Rahman[3], Nabeel Mohammed[1], Shafin Rahman[1]

[1]Machine Intelligence Lab (MILab), North South University, Dhaka, Bangladesh
[2]Visa Inc., Atlanta, GA, USA
[3]Apurba Technologies, Sunnyvale, CA, USA

Abstract

This paper describes our participation in Task 1A and Task 1B of the BLP Workshop[1], focused on Bangla Multi-task Hatespeech Identification. Our approach involves systematic evaluation of four transformer models: BanglaBERT, XLM-RoBERTa, IndicBERT, and Bengali-Abusive-MuRIL. To enhance performance, we implemented an ensemble strategy that averages output probabilities from these transformer models, which consistently outperformed individual models across both tasks. The baseline classical methods demonstrated limitations in capturing complex linguistic cues, underscoring the superiority of transformer-based approaches for low-resource hate speech detection. Our solution initially achieved F1 scores of 0.7235 (ranked 12th) for Task 1A and 0.6981 (ranked 17th) for Task 1B among participating teams. Through post-competition refinements, we improved our Task 1B performance to 0.7331, demonstrating the effectiveness of ensemble methods in Bangla hate speech detection.

1 Introduction

In recent years, online communication has become a primary medium for individuals to express opinions and emotions. With the growing use of digital platforms, the prevalence of hate speech has also increased rapidly. Hate speech refers to language that spreads hostility or discrimination against individuals or groups based on attributes such as appearance, religion, ethnicity, or gender (Papcunová et al., 2023b). Such content not only fuels social conflict but can also damage international relations and, in extreme cases, contribute to violent outcomes, including wars (Sahoo et al., 2024). While significant progress has been made in detecting hate speech in high-resource languages such as English (MacAvaney et al., 2019; Kearns et al., 2023), the challenge remains particularly acute for underrepresented languages like Bangla, where datasets, resources, and detection systems are still scarce.

Existing approaches to hate speech detection often rely on classical machine learning algorithms (Mullah and Zainon, 2021; Subramanian et al., 2023), which struggle to capture the linguistic nuances present in hateful texts, especially in low-resource languages. Moreover, most prior studies focus on binary classification (Subramanian et al., 2023), distinguishing hate from non-hate, without addressing the finer-grained categorization of hate into types such as abusive, religious, political, or sexist. Equally overlooked is the identification of the target of hate, whether directed toward individuals, organizations, communities, or society. These limitations highlight a major gap in comprehensive Bangla hate speech identification.

To address these gaps, this study advances beyond binary hate speech detection and tackles two key tasks: ***(a)*** fine-grained classification of hate speech types and ***(b)*** identification of the targeted group. As a baseline, we experimented with classical machine learning classifiers using sentence embeddings from a pretrained sentence transformer, but their limitations in capturing complex linguistic cues underscored the need for transformer-based approaches. We therefore employed four Bangla-specific models, BanglaBERT (Hasan et al., 2020), XLM-RoBERTa (Conneau et al., 2020), IndicBERT (Kakwani et al., 2020), and Bengali-Abusive-MuRIL (Das et al., 2022), and further enhanced performance through an ensemble strategy that averages their output probabilities. The ensemble consistently outperformed individual models across both tasks, demonstrating its effectiveness for Bangla hate speech detection.

The main contributions are as follows: ***(i)*** Development of a Bangla hate speech detection framework that extends beyond binary classification to perform fine-grained hate categorization and target

[1] https://multihate.github.io/

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 461–467
December 23, 2025 ©2025 Association for Computational Linguistics

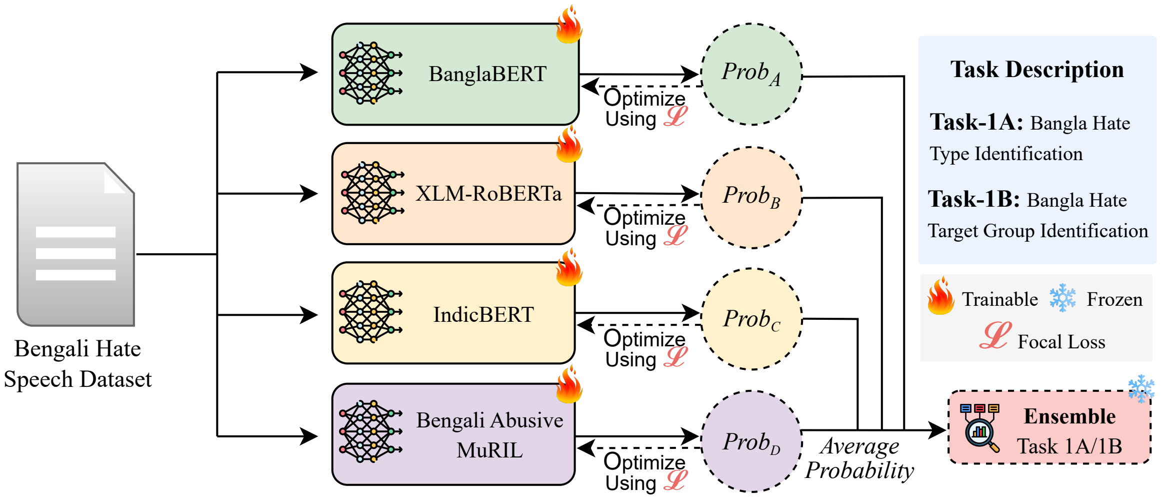


Figure 1: Overview of the proposed Bangla hate speech type identification and Bangla Hate speech target group identification framework. Bangla text inputs are passed through four transformer-based models (BanglaBERT, XLM-RoBERTa, IndicBert, and Bengali-Abusive-MuRIL). A focal loss function is applied to address class imbalance, and the averaged outputs are used for final classification and target group identification.

group identification. ***(ii)*** Systematic evaluation of baseline and advanced approaches, where machine learning classifiers are compared against Bangla-specific transformer models, highlighting the limitations of traditional methods. ***(iii)*** Introduction of a transformer model ensemble that achieves superior performance across both tasks, demonstrating the effectiveness of ensembling in a low-resource language setting.

2 Related Works

Hate speech detection has been widely studied in recent years, with early surveys highlighting challenges such as linguistic subtlety, implicit hate, and limited annotated resources (MacAvaney et al., 2019). While most research has focused on high-resource languages, recent studies emphasize the urgent need for progress in low-resource languages, where annotated corpora and robust models are scarce (Das et al., 2024). For Bangla, initial resources such as BD-SHS (Romim et al., 2022) and BanglaHateBERT (Jahan et al., 2022) have facilitated the development of benchmark systems. Transformer-based approaches have proven effective for hate speech detection (Chakravorty et al., 2024). Monolingual models such as MahaBERT and BanglaBERT often outperform multilingual baselines in capturing language-specific nuances, while multilingual models like MuRIL and XLM-RoBERTa demonstrate strong cross-lingual transfer (Ghosh and Senapati, 2022, 2025). Recent studies further demonstrate that multilingual and multi-task learning can improve generalization across domains and targets of hate (Yuan and Rizoiu, 2025). Generative large language models (LLMs) also show promising results, surpassing traditional transformer baselines in Bangla hate detection tasks (Faria et al., 2024). Finally, datasets such as IndicCONAN (Sahoo et al., 2024) support counter-narrative generation, broadening the scope of hate speech research in Indic languages. Together, these works highlight the evolution from classical methods (Kearns et al., 2023; Papcunová et al., 2023a) to transformer and LLM-driven approaches, underscoring the importance of developing robust systems for Bangla and other low-resource languages. While prior works highlight individual model strengths, we show that an ensemble of state-of-the-art transformers yields more robust and accurate fine-grained Bangla hate speech detection.

3 Methods

Problem Formulation: Given a dataset $\mathcal{D} = \{(T_i, y_i)\}_{i=1}^{N}$ of Bangla text samples T_i and their corresponding labels y_i, our objectives are: ***(i)*** *Hate speech type identification*, where $y_i \in \{1, \dots, 6\}$ denotes one of six predefined hate speech categories, and ***(ii)*** *Target group identification*, where $y_i \in \{1, \dots, 5\}$ denotes the specific target group five possible categories.

Solution Strategy: Bangla hate speech detection is particularly challenging due to limited resources and significant class imbalance. To address these issues, we adopt the following strategy: ***(a) Multi-***

Model Ensemble: Four transformer-based models are fine-tuned on dataset $\mathcal{D}$ to capture diverse linguistic features. ***(b) Focal Loss Optimization:*** A focal loss function is employed during training to mitigate class imbalance by emphasizing harder, misclassified samples. ***(c) Prediction Aggregation:*** The output probabilities of the four models are averaged to obtain robust final predictions for both hate category and target group identification.

3.1 Revisiting Transformer Models

Bangla hate speech detection is challenging due to linguistic variations, dialect diversity, and code-mixing. As baselines, we fine-tune four distinct transformer-based models on the dataset. **BanglaBERT** is a monolingual model trained on a large Bangla corpus. **XLM-RoBERTa** is a multilingual model that captures cross-lingual representations. **IndicBERT** is trained on multiple Indic languages, including Bangla, and leverages shared linguistic features. **Bengali-Abusive-MuRIL** is pretrained with a focus on abusive content, making it relevant for hate speech detection.

3.2 Ensemble Approach

As Bangla hate speech detection is a complex task due to its low-resource nature, individual models often fail to capture all nuances. Relying on a single model often fails to capture these nuances. To overcome this, we employ an ensemble of four transformer-based models: Bangla BERT, XLM-RoBERTa, IndicBERT, and Bengali-Abusive-MuRIL. Each model is fine-tuned individually on the dataset to learn task-specific features while preserving its pretraining knowledge. As illustrated in Figure 1, the input text is processed in parallel through the four models, and their predictions are later combined. The ensemble strategy leverages complementary strengths of different pretrained models, thereby improving generalization and robustness compared to any single model.

Focal Loss Optimization: The dataset (Hasan et al., 2025b) is highly imbalanced shown in Appendix A.1, with hateful instances being significantly underrepresented. Training with standard cross-entropy loss leads to models biased toward the majority (non-hateful) class. To mitigate this, we adopt focal loss, defined as:

$$\mathcal{L}_{\text{focal}}(p_t) = -\alpha(1 - p_t)^{\gamma} \log(p_t)$$

where p_t is the predicted probability for the true class, α is a balancing factor, and γ is a focusing parameter. The modulating term $(1-p_t)^{\gamma}$ reduces the relative loss for well-classified examples, thus placing more emphasis on harder, misclassified samples. This helps the model pay more attention to minority classes and subtle hate speech instances.

Prediction Aggregation: After training, the outputs of each model are combined into a single prediction. We adopt a simple yet effective strategy of averaging all predicted probabilities:

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$$

where M is the number of models and $\hat{y}_m$ is the output probability from the m-th model. Averaging stabilizes predictions, reduces variance, and avoids overfitting that may arise from a single model.

4 Experiments

4.1 Setup

Dataset: We conducted our experiments on the BLP Shared Task 1A and Task 1B datasets (Hasan et al., 2025a,b) for Bangla hate speech. Each dataset consists of 35,522 training samples, 2,512 validation samples, 2,512 dev-test samples, and 10,200 test samples. Task 1A focuses on classifying the type of hate speech across six categories: None, Abusive, Political Hate, Religious Hate, Sexism, and Profane. Task 1B, on the other hand, involves identifying the target group with five classes: None, Individual, Organization, Community, and Society. The detailed class-wise distributions across all dataset splits are provided in Appendix A.1.

Evaluation Metrics: Performance was assessed using Precision (P), Recall (R), and F1-score, reported on both the dev-test and test datasets for both Task 1A and Task 1B.

4.2 Main Results

Table 1 presents the performance of individual models and the ensemble system on both subtasks: Task 1A (Bangla hate Speech Type Identification) and Task 1B (Bangla hate Speech Target Group Identification) for the dev-test and test datasets.

For **Task 1A**, BanglaBERT and Bengali-Abusive-MuRIL achieved strong results, with F1-scores of 0.7312 and 0.7075 on the dev-test set, respectively. IndicBERT performed moderately with an F1 score of 0.6455, while XLM-RoBERTa lagged behind, achieving only 0.4211 F1 on the

Model	Task 1A:Bangla Hate Speech Type Identification						Task 1B:Bangla Hate Speech Target Group Identification					
	Dev-Test			Test			Dev-Test			Test		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
BanglaBERT (Hasan et al., 2020)	0.7385	0.7269	0.7312	0.7178	0.7046	0.7092	0.7399	0.7186	0.7259	0.7320	0.7123	0.7187
XLM-RoBERTa (Conneau et al., 2020)	0.3318	0.5760	0.4211	0.3179	0.5638	0.4066	0.3700	0.6083	0.4601	0.3568	0.5974	0.4468
IndicBERT (Kakwani et al., 2020)	0.6446	0.6525	0.6455	0.6364	0.6481	0.6393	0.6619	0.6712	0.6644	0.6491	0.6612	0.6526
Bengali-Abusive-MuRIL (Das et al., 2022)	0.7058	0.7109	0.7075	0.6978	0.7019	0.6996	0.7288	0.7277	0.7275	0.7068	0.7081	0.7070
Ensemble of All (with focal loss)	**0.7358**	**0.7448**	**0.7396**	**0.7211**	**0.7271**	**0.7235**	**0.7447**	**0.7464**	**0.7444**	**0.7320**	**0.7350**	**0.7331**

Table 1: Performance of individual models and the ensemble on Task 1A and Task 1B. Results are reported on dev-test and test sets in terms of Precision (P), Recall (R), and F1-score. Best results are in **bold**, and the second-best are underlined. More detailed ensembling results are presented in Appendix A.2.

dev-test. The ensemble of the four models with focal loss outperformed all individual models, achieving the best F1-scores of 0.7396 on the dev-test and 0.7235 on the test set. For **Task 1B**, a similar trend was observed. BanglaBERT and Bengali-Abusive-MuRIL were competitive baselines, with F1-scores of 0.7259 and 0.7275 on the dev-test, respectively. IndicBERT performed slightly lower, and XLM-RoBERTa again underperformed compared to monolingual and multilingual models tailored for Indic languages. Our proposed ensemble achieved the highest overall performance, reaching an F1 score of 0.7444 on the dev-test and 0.7331 on the test set.

These results demonstrate that while Bangla-specific and Indic-focused models are strong baselines for hate speech detection, the ensemble strategy with focal loss provides consistent gains across both subtasks, highlighting the benefits of combining diverse model predictions.

4.3 Error Analysis

Figure 2 shows error patterns across both tasks, closely tied to the dataset distribution (Table 2 in Appendix A.1). In Task 1A, most errors occur in Abusive (1,088 errors) and Political Hate (533 errors), the largest hate-related categories, where greater lexical diversity makes classification harder. In contrast, Religious Hate (28 errors) and Sexism (89 errors) appear better handled, though this is partly due to their small sample sizes (676 and 122 samples), which limit variability rather than stronger generalization. Profane shows moderate difficulty with 163 errors.

For Task 1B, the None class dominates errors (929 errors), reflecting its overwhelming size (21,190 samples) and the challenge of distinguishing non-targeted from subtly targeted text. Other categories, including Individual (555 errors), Organization (470 errors), and Society (359 errors), show comparable difficulty, while Community (389 errors) is relatively more stable despite fewer examples. Overall, class imbalance drives most errors, with frequent misclassifications in majority classes and limited coverage for minority ones.

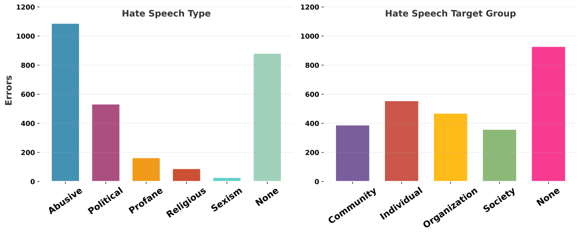


Figure 2: Error distribution across classification tasks. The bar charts show the number of misclassified samples for each class in Task 1A (left) and Task 1B (right).

4.4 Discussion

Our findings show that Bangla-specific models such as BanglaBERT and Bengali-Abusive-MuRIL outperform general multilingual models like XLM-RoBERTa, emphasizing the importance of language-focused pretraining. IndicBERT achieved moderate results, but its multilingual nature limited its effectiveness compared to Bangla-focused models. Across both subtasks, the ensemble approach consistently showed the best scores. These results highlight the value of combining diverse models to enhance generalization in Bangla hate speech detection.

Conclusion

In this study, we presented our proposed method and results on the BLP Shared Task 1A (Bangla hate speech type classification) and Task 1B (target group identification). Our work highlights the importance of tackling the challenging problem of hate speech detection in a low-resource language (Bangla), particularly in identifying both the type of hate speech and its target group. To this end, we employed an ensemble of four transformer-based models, demonstrating the effectiveness of robust NLP systems in mitigating harmful online content.

Limitations

Although our ensemble framework achieved competitive rankings in the shared task, it faced notable constraints. The reliance on pretrained transformer models introduced high computational costs during fine-tuning, which may not be feasible in resource-limited environments. Furthermore, class imbalance in the dataset, particularly for underrepresented categories such as Sexism and Religious Hate, limited the models ability to generalize across all classes. These challenges contributed to misclassifications observed in the error analysis, highlighting difficulties in handling overlapping or subtle linguistic cues. Another limitation is that the ensemble approach, while effective, increases inference time compared to single-model systems, which could hinder real-time or large-scale deployment. Moreover, our ensembling method used uniform averaging, which may not optimally capture the varying strengths of individual models.

Future Works: Building on the competition results, future work can focus on improving class balance through techniques such as data augmentation, resampling strategies, or more adaptive loss functions. Exploring alternative ensemble strategies beyond simple probability averaging, such as weighted ensembling or stacking, could further enhance performance by leveraging model-specific strengths. Finally, incorporating more efficient fine-tuning methods (e.g., parameter-efficient tuning) may reduce computational demands, enabling broader participation in similar low-resource shared tasks while maintaining strong performance.

References

Debamalya Chakravorty, Arijit Das, and Diganta Saha. 2024. Multilingual hate speech detection using transformer-based deep learning approaches. In *2024 11th International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 126–131.

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.

Mithun Das, Somnath Banerjee, and Animesh Mukherjee. 2022. Data bootstrapping approaches to improve low resource abusive language detection for indic languages. In *Proceedings of the 33rd ACM conference on hypertext and social media*, pages 32–42.

Susmita Das, Arpita Dutta, Kingshuk Roy, Abir Mondal, and Arnab Mukhopadhyay. 2024. A survey on automatic online hate speech detection in low-resource languages. arXiv preprint arXiv:2411.19017.

Fatema Tuj Johora Faria, Laith H. Baniata, and Sangwoo Kang. 2024. Investigating the predominance of large language models in low-resource bangla language over transformer models for hate speech detection: A comparative analysis. *Mathematics*, 12(23):3687.

Koyel Ghosh and Apurbalal Senapati. 2022. Hate speech detection: a comparison of mono and multilingual transformer model with cross-language evaluation. In *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation (PACLIC)*, pages 853–865, Manila, Philippines. Association for Computational Linguistics.

Koyel Ghosh and Apurbalal Senapati. 2025. Hate speech detection in low-resourced indian languages: An analysis of transformer-based monolingual and multilingual models with cross-lingual experiments. *Natural Language Processing*, 31(2):393–414.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M. Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2612–2623, Online. Association for Computational Linguistics.

M. S. Jahan, M. Haque, N. Arhab, and M. Oussalah. 2022. BanglaHateBERT: Bert for abusive language detection in bengali. In *Proceedings of the Second Workshop on Abusive Language Online (co-located with LREC 2022)*.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul NC, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020. Indicnlpsuite: 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.

Colm Kearns, Gary Sinclair, Jack Black, Mark Doidge, Thomas Fletcher, Daniel Kilvington, Katie Liston, Theo Lynn, and Pierangelo Rosati. 2023. A scoping review of research on online hate and sport. *Communication & Sport*, 11(2):402–430.

Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. 2019. Hate speech detection: Challenges and solutions. *PloS one*, 14(8):e0221152.

Nanlir Sallau Mullah and Wan Mohd Nazmee Wan Zainon. 2021. Advances in machine learning algorithms for hate speech detection in social media: a review. *IEEE access*, 9:88364–88376.

Jana Papcunová, Marcel Martončík, Denisa Fedáková, Michal Kentoš, and Matúš Adamkovič. 2023a. Perception of hate speech by the public and experts: Insights into predictors of the perceived hate speech towards migrants. *Cyberpsychology, Behavior, and Social Networking*, 26:546–553.

Jana Papcunová, Marcel Martončik, Denisa Fedáková, Michal Kentoš, Miroslava Bozogáňová, Ivan Srba, Robert Moro, Matúš Pikuliak, Marián Šimko, and Matúš Adamkovič. 2023b. Hate speech operationalization: a preliminary examination of hate speech indicators and their structure. *Complex & intelligent systems*, 9(3):2827–2842.

N. Romim, M. Ahmed, M. S. Islam, A. S. Sharma, H. Talukder, and M. R. Amin. 2022. Bd-shs: A benchmark dataset for learning to detect online bangla hate speech in different social contexts. In *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, pages 5153–5162.

Nihar Ranja Sahoo, Gyana Prakash Beria, and Pushpak Bhattacharyya. 2024. Indicconan: A multilingual dataset for combating hate speech in indian context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22313–22321.

Malliga Subramanian, Veerappampalayam Easwaramoorthy Sathiskumar, G Deepalakshmi, Jaehyuk Cho, and G Manikandan. 2023. A survey on hate speech detection and sentiment analysis using machine learning and deep learning models. *Alexandria Engineering Journal*, 80:110–121.

Lanqin Yuan and Marian-Andrei Rizoiu. 2025. Generalizing hate speech detection using multi-task learning: A case study of political public figures. *Computer Speech & Language*, 89:101690.

A Appendix

A.1 Dataset Distribution Details

Table 2 provides a detailed class-wise breakdown of the dataset used in our experiments, covering both Task 1A (hate speech type classification) and Task 1B (target group identification). The dataset was split into four partitions: training, validation, dev-test, and test.

Task 1A: Hate Speech Type

Class	Train	Val	Dev-Test	Test
None	19,954	1,451	1,447	5,751
Abusive	8,212	564	549	2,312
Political Hate	4,227	291	283	1,220
Religious Hate	676	38	40	179
Sexism	122	11	8	29
Profane	2,331	157	185	709
Total	**35,522**	**2,512**	**2,512**	**10,200**

Task 1B: Target Identification

Class	Train	Val	Dev-Test	Test
None	21,190	1,536	1,528	6,093
Individual	5,646	364	391	1,571
Organization	3,846	292	292	1,152
Community	2,635	179	159	759
Society	2,205	141	142	625
Total	**35,522**	**2,512**	**2,512**	**10,200**

Table 2: Dataset distribution showing class-wise breakdown for Task 1A (Bangla hate speech type classification) and Task 1B (Bangla hate speec target group identification) across all splits.

Bangla Hate Speech Type Classification: This task focuses on categorizing each instance into one of six classes: *None*, *Abusive*, *Political Hate*, *Religious Hate*, *Sexism*, and *Profane*. The distribution is highly imbalanced, with the majority of samples belonging to the *None* and *Abusive* categories. Specifically, the training set contains 19,954 instances of *None* and 8,212 instances of *Abusive*, compared to only 122 samples of *Sexism*. Such imbalance poses challenges for model training, as minority classes like *Sexism* and *Religious Hate* are be underrepresented.

Bangla Hate Speech Target Group Identification: This task aims to identify the target group of the hateful expression, categorized into five groups: *None*, *Individual*, *Organization*, *Community*, and *Society*. As shown in Table 2, the distribution is again skewed, with *None* being the dominant category (21,190 training instances), followed by *Individual* (5,646 samples) and *Organization* (3,846 samples). The minority categories, such as *Community* (2,635 samples) and *Society* (2,205 samples),

Model / Ensemble	Task 1A: Bangla Hate Speech Type Identification						Task 1B: Bangla Hate Speech Target Group Identification					
	Dev-Test			Test			Dev-Test			Test		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
2-Model Ensembles												
BanglaBERT + XLM-RoBERTa	0.7202	0.7237	0.7201	0.7049	0.7071	0.7034	0.7464	0.7392	0.7415	0.7005	0.6937	0.6305
BanglaBERT + IndicBERT	0.7022	0.7133	0.7051	0.6874	0.6994	0.6909	0.7334	0.7336	0.7312	0.7203	0.7216	0.7182
BanglaBERT + B-A-MuRIL	0.7166	0.7233	0.7192	0.7054	0.7107	0.7077	0.7417	0.7408	0.7399	0.7210	0.7212	0.7200
XLM-RoBERTa + IndicBERT	0.6318	0.6524	0.6348	0.6310	0.6507	0.6319	0.7259	0.7328	0.7285	0.5786	0.6437	0.5491
XLM-RoBERTa + B-A-MuRIL	0.7065	0.7149	0.7097	0.6958	0.7021	0.6986	0.7391	0.7412	0.7397	0.6795	0.6912	0.6385
IndicBERT + B-A-MuRIL	0.7040	0.7169	0.7082	0.6835	0.6965	0.7088	0.7006	0.7320	0.7261	0.7052	0.7165	0.7087
3-Model Ensembles												
BanglaBERT + XLM-RoBERTa + IndicBERT	0.6865	0.7042	0.6868	0.6774	0.6950	0.6768	0.6979	0.7165	0.6832	0.6929	0.7084	0.6734
BanglaBERT + XLM-RoBERTa + B-A-MuRIL	0.7210	0.7336	0.7256	0.7068	0.7186	0.7109	0.7320	0.7452	0.7308	0.7141	0.7301	0.7138
BanglaBERT + IndicBERT + B-A-MuRIL	0.7201	0.7289	0.7237	0.7159	0.7229	0.7190	0.7385	0.7416	0.7382	0.7243	0.7291	0.7248
XLM-RoBERTa + IndicBERT + B-A-MuRIL	0.6950	0.7125	0.6936	0.6770	0.6977	0.6763	0.7029	0.7221	0.6901	0.6855	0.7059	0.6712
Ensemble (with focal loss)	**0.7358**	**0.7448**	**0.7396**	**0.7211**	**0.7271**	**0.7235**	**0.7447**	**0.7464**	**0.7444**	**0.7320**	**0.7350**	**0.7331**

Table 3: Performance comparison of two-model ensembles, three-model ensembles, and the final four-model ensemble on Task 1A and Task 1B. Results are reported in terms of Precision (P), Recall (R), and F1-score. Best results are in **bold**. B-A-MuRIL denotes Bengali-Abusive-MuRIL.

contain relatively fewer examples.

A.2 Detailed Results of Model and Ensemble Experiments

Table 3 presents the complete experimental results for both sub-tasks: **Task 1A** (Bangla Hate Speech Type Identification) and **Task 1B** (Bangla Hate Speech Target Group Identification). We report Precision (P), Recall (R), and F1-score on both the **dev-test** and **test** splits for all two-model ensembles, three-model ensembles, and the final four-model ensemble trained with focal loss.

Two-Model Ensembles: Pairwise ensembles demonstrate varying levels of performance depending on model combinations. For Task 1A, **BanglaBERT + Bengali-Abusive-MuRIL** achieves the strongest results among 2-model ensembles with F1-scores of 0.7192 (dev-test) and 0.7077 (test). The **BanglaBERT + XLM-RoBERTa** combination yields comparable dev-test performance (0.7201 F1) but slightly lower test scores (0.7034 F1). In Task 1B, **BanglaBERT + XLM-RoBERTa** shows strong dev-test performance with an F1 score of 0.7415, though it experiences a notable drop on the test set (0.6305 F1). The most stable 2-model ensemble for Task 1B is **BanglaBERT + Bengali-Abusive-MuRIL**, achieving 0.7399 (dev-test) and 0.7200 (test). Ensembles involving XLM-RoBERTa without BanglaBERT show weaker performance, particularly **XLM-RoBERTa + IndicBERT**, which achieves only 0.6348 F1 (dev-test) and 0.6319 F1 (test) in Task 1A.

Three-Model Ensembles: Three-way ensembles demonstrate improved stability and performance over most pairwise combinations. The **BanglaBERT + IndicBERT + Bengali-Abusive-MuRIL** ensemble achieves the highest performance among this group, with F1-scores of 0.7237 (dev-test) and 0.7190 (test) in Task 1A, and 0.7382 (dev-test) and 0.7248 (test) in Task 1B. The **BanglaBERT + XLM-RoBERTa + Bengali-Abusive-MuRIL** combination also performs strongly, achieving 0.7256 (dev-test) and 0.7109 (test) in Task 1A, and 0.7308 (dev-test) and 0.7138 (test) in Task 1B. These results indicate that combining Bangla-specific pretrained models with multilingual counterparts helps capture diverse linguistic cues while maintaining robustness. Ensembles excluding BanglaBERT consistently underperform, with **XLM-RoBERTa + IndicBERT + Bengali-Abusive-MuRIL** achieving only 0.6936 F1 (dev-test) in Task 1A.

Final Four-Model Ensemble with Focal Loss: The strongest performance is obtained with the final **all-model ensemble** (BanglaBERT, XLM-RoBERTa, IndicBERT, and Bengali-Abusive-MuRIL) trained using focal loss. This configuration achieves **0.7396 F1 (dev-test) and 0.7235 F1 (test) on Task 1A**, and **0.7444 F1 (dev-test) and 0.7331 F1 (test) on Task 1B**, outperforming all other ensemble configurations. Notably, this represents improvements of 0.0159 F1 points (dev-test) and 0.0045 F1 points (test) over the best 3-model ensemble in Task 1A, and 0.0062 F1 points (dev-test) and 0.0083 F1 points (test) in Task 1B. The final ensemble also demonstrates better generalization, with precision of 0.7358 and recall of 0.7448 on the dev-test for Task 1A, indicating a balanced approach to both false positives and false negatives.

Catalyst at BLP-2025 Task 1: Transformer Ensembles and Multi-task Learning Approaches for Bangla Hate Speech Detection

Nahid Hasan
Department of Computer Science and Engineering
Rajshahi University of Engineering & Technology
Rajshahi, Bangladesh
nahidhasan2003131@gmail.com

Abstract

We present a compact, cost-efficient system for the BLP-2025 Bangla Multi-task Hate Speech Identification Task 1, which requires fine-grained predictions across three dimensions: type, target, and severity. Our method pairs strong multilingual transformer encoders with two lightweight strategies: task-appropriate ensembling to stabilize decisions across seeds and backbones, and a multi-task head that shares representations while tailoring outputs to each subtask. As Catalyst, we ranked **7th** on Subtask 1A with micro-F1 **73.05**, **8th** on Subtask 1B with **72.79**, and **10th** on Subtask 1C with **72.40**. Despite minimal engineering, careful model selection and straightforward combination rules yield competitive performance and more reliable behavior on minority labels. Ablations show consistent robustness gains from ensembling, while the multi-task head reduces cross-dimension inconsistencies. Error analysis highlights persistent challenges with code-mixed slang, implicit hate, and target ambiguity, motivating domain-adaptive pretraining and improved normalization.

1 Introduction

The fast increase in the number of social media platforms has resulted in a growing concern regarding the harmful online content, especially hate speech in under-resourced languages such as Bangla. Informal language, code-mixing, and culturally sensitive hate utterances make it difficult to detect such content. Although the results of multilingual transformers and domain-adaptive pretraining are promising, they are vulnerable to major code-mixing and distributional drift (Sharif et al., 2021; Caselli et al., 2021).

The Bangla Multi-task Hate Speech Identification shared task (Hasan et al., 2025b) attempts to resolve these issues by proposing a complex framework that does not just deal in binary classification. It lays stress on the practical moderation requirements and promotes systems, which manage delicate phenomena by making organized forecasts along associated dimensions. Our work fills the gaps left by traditional hate speech detection which typically uses binary classification that does not provide the subtlety needed to do content moderation effectively. The common task helps develop the field because it involves systems conducting fine-grained analysis on three dimensions: type of hate, target, and level of severity. In the case of BLP-2025 Task 1, we trained a system that is a combination of transformer-based models and ensemble methods and multi-task learning. We experimented with several multilingual language models and developed simple yet efficient strategies for each subtask, leveraging both cross-lingual transfer and effective ensembling techniques.. Our method shows that basic combinations and good models can be highly effective with simple feature engineering, which confirms results that well-planned model combinations can be more reliable than multicomponent models with more advanced feature engineering designs (Plaza-Del-Arco et al., 2021; Saha et al., 2021).

The main contributions of our work include:

- An extensive analysis of four multilingual hate speech transformer models in Bangla.
- Good ensemble strategies that enhance performance by means of model combination.
- A multi-task learning system that deals with several classification goals at once.
- Competitive outcomes in all three subtasks that included practical and efficient solutions.

Our findings indicate that carefully designed transformer architectures, paired with suitable training strategies, have the ability to deal with the challenges of hate speech detection in Bangla social

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 468–475
December 23, 2025 ©2025 Association for Computational Linguistics

media whilst retaining enough simplicity to be deployable in practice.

2 Related Work

Transformer-based encoders have now become the standard method of abusive and hate speech detection, particularly in Indic code-mixed environments where multilingual models are significantly more effective than classical ML and RNN models of abusive speech detection (Sharif et al., 2021). Ensembling also increases stability and precision, and genetic-algorithm-weighted mixtures of transformer runs achieve the top scores in Dravidian-LangTech 2021 and similar analogous score increases of similar magnitude are reported for Arabic (Saha et al., 2021; de Paula et al., 2025). Multitask learning that simultaneously predicts hate or aggression and affective cues like sentiment and emotion has the added advantage of sharing a transformer backbone but having task-specific heads (Plaza-Del-Arco et al., 2021). Domain-adaptive pretraining is also significant: HateBERT, which is additionally trained on abuse-rich data on Reddit, performs better and has stronger cross dataset portability than vanilla BERT (Caselli et al., 2021). Finally, multilingual transformers provide strong cross-lingual transfer, motivating the use of few-shot generalization and transfer-aware training to better handle Bangla and code-mixed text (Ni et al., 2020).

3 Task Description

This study introduces the Bangla Multi-Task Hate Speech Identification shared task[1], which represents a substantial step beyond the traditional binary-classification paradigm toward a more fine-grained, three-dimensional detection framework. In this shared task, there are three subtasks.

- **Subtask 1A:** Given a Bangla text collected from YouTube comments, classify the text as abusive, sexism, religious hate, political hate, profane, or none.
- **Subtask 1B:** Using a Bangla text gathered from the YouTube comments, classify the hate directed towards individuals, organizations, communities, or society.
- **Subtask 1C:** It is a multi-task arrangement. Based on a Bangla text acquired in YouTube comments, classify it by type of hate, severity, and targeted group.

Subtask	Label	Example
1A	Political Hate	আওয়ামী লীগের সন্ত্রাসী কবে দরবেন এই সাহস আপনাদের নাই
1A	Abusive	শালায় এক নাম্বার বাটপার
1B	Organization	খানকির পোলারা হেডার নিউজ দেস নিউজের মাঝখানে দুইবার দেস এড
1B	None	আমার মতো কমেন্ট পরতে ভালোবাসো কারা
1C	Political Hate, Severe, Organization	আপনারা জুতা খাওয়া পাটি

Table 1: Examples by subtask with labels.

3.1 Dataset Description

The dataset (Hasan et al., 2025a) utilized to carry out this task consists of YouTube comments about socially sensitive matters within the Bangla-speaking region. The remarks are in Bangla and show the informal, noisy nature of user-generated content, such as spelling variability, code-mixing and colloquialism. All samples are labeled with the three classification objectives that are related to the subtasks. In the real world hate speech is messy. This is represented in the dataset by its non-uniform categories and a thin line between aggressive political expression and actual hate, and it represents a real test of the subtlety and expertise of a model. Table 1 demonstrates representative examples from the dataset, while Table 2 provides statistical overview.

Subtask	Train	Dev-Test	Test
1A: Hate Type	35,522	2,512	10,200
1B: Target	35,522	2,512	10,200
1C: Multi-task	35,522	2,512	10,200

Table 2: Dataset statistics across all three subtasks.

4 System Description

The proposed system of the hate speech identification shared task utilizes both single-task classification with transformer-based ensembles and multi-task learning with joint prediction. Our approach is a fine-tuning process, in which multilingual transformers trained on general data are fine-tuned to the specific data and ensembled using ensemble techniques. Figure 1 shows the overall system architecture.

[1] https://github.com/AridHasan/blp25_task1

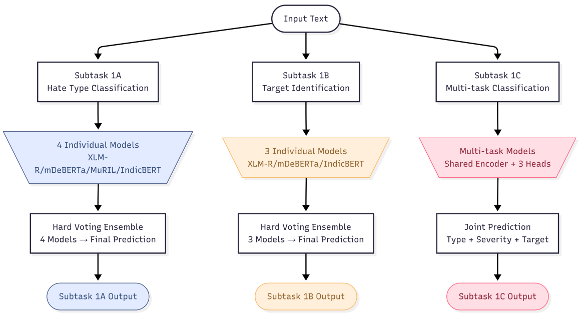


Figure 1: Overall system architecture showing separate pipelines for each subtask with ensemble strategies for 1A/1B and multi-task learning for 1C.

4.1 Models

We employed four cross-lingual Transformer encoders to take advantage of cross-lingual transfer to Bangla and with coverage of complementary architectural options and pretraining corpora.

- **XLM-RoBERTa-large** (Conneau et al., 2020) is chosen due to its pretraining on 100 languages allowing a high cross-lingual transfer and an adequate coverage of the subwords in Bangla.
- **microsoft/mdeberta-v3-base** (He et al., 2023) is selected because it implements disentangled attention in an efficient backbone, providing an appropriate trade-off between accuracy and efficiency in our context.
- **google/muril-base-cased** (Khanuja et al., 2021) is included since it is specifically trained for Indian languages and is therefore well suited to regional scripts, code-mixing, and transliteration phenomena.
- **ai4bharat/IndicBERTv2-MLM-only** (Doddapaneni et al., 2023) is included since its tokenizer and vocabulary is optimized to Indic scripts, aiding in the capturing of South Asian morphological and orthographic aspects.

All fine-tuning experiments for the subtasks were conducted on Google Colab using single NVIDIA T4 GPU. The detailed hyperparameters and training configurations are provided in Appendix A, and the complete source code is available in our public repository[2].

4.2 Subtask 1A: Hate Type Classification

Subtask 1A required assigning each Bangla YouTube comment to one of six categories, *abusive*, *sexism*, *religious hate*, *political hate*, *profane*, or *none*. We fine-tuned four multilingual encoders, *IndicBERTv2*, *XLM-RoBERTa-large*, *mDeBERTa-v3-base*, and *MuRIL-base*, and aggregated predictions via hard voting to stabilize decisions across architectures and seeds. For an instance x_i with model set $\mathcal{M}$ and label set $\mathcal{Y}$, the ensemble prediction was

$$\hat{y}_i = \arg\max_{y \in \mathcal{Y}} \sum_{m \in \mathcal{M}} \mathbf{1}\Big\{y = \hat{y}_i^{(m)}\Big\}.$$

This rule yielded a consistent improvement over the strongest single model (Table 3), indicating complementary inductive biases among encoders and increased robustness to label ambiguity.

4.3 Subtask 1B: Target Identification

Subtask 1B identified the target of the hateful content.We fine-tuned three encoders, *XLM-RoBERTa-large*, *mDeBERTa-v3-base*, and *IndicBERTv2*, and applied the same hard voting rule as in Subtask 1A, with $\mathcal{M}$ restricted to these three models. The ensemble outperformed the strongest individual model (Table 3), improving reliability on distinctions among *individual*, *organization*, *community*, and *society* targets.

4.4 Subtask 1C: Multi-task Classification

Our multi-task architecture used a shared encoder with dedicated classification heads for Hate Type, Severity, and Target prediction.We evaluated *IndicBERTv2*, *XLM-RoBERTa-base*, and *mDeBERTa-v3-base* as the shared backbone. Figure 2 illustrates the shared-encoder setup and the three task-specific heads. Each subtask produced an independent cross-entropy loss $\mathcal{L}$, and the overall objective was

$$\mathcal{L}_{total} = \mathcal{L}_{type} + \mathcal{L}_{severity} + \mathcal{L}_{target}.$$

During backpropagation, gradients from all subtasks jointly updated the shared encoder, enabling the model to learn common linguistic and semantic representations across hate-related dimensions. We formed the sentence representation from the encoder's [CLS] token with dropout before the task heads. No explicit loss weighting was applied. Equal contributions from all subtasks yielded stable convergence without noticeable task interfer-

[2] https://github.com/nahid2003131/blp25_task1_catalyst

ence. The *IndicBERTv2* variant performed best and was selected as the final submission.

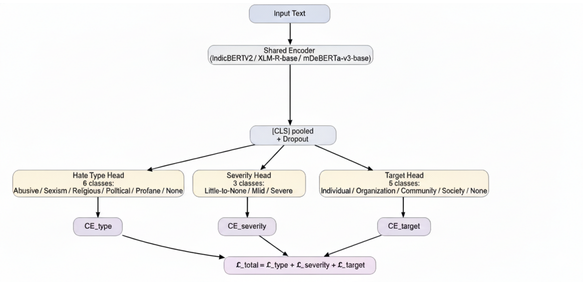

Input Text
Shared Encoder
(IndicBERTV2 / XLM-R-base / mDeBERTa-v3-base)
[CLS] pooled
+ Dropout
Hate Type Head
6 classes:
Abusive / Sexism / Relgious / Poltical / Profane / None
Severity Head
3 classes:
Little-to-None / Mlid / Severe
Target Head
5 classes:
Individual / Organization / Community / Society / None
CE_type
CE_severity
CE_target
£_total = £_type + £_severity + £_target

Figure 2: Multi-task learning architecture for Subtask 1C.

5 Results and Findings

The micro-F1 scores on development-test and official test sets are summarized in Table 3 with respect to all three subtasks. We compare individual transformer model performances with our ensemble approaches for Subtasks 1A and 1B, and evaluate multi-task learning models for Subtask 1C.The final column displays our official rankings on the leaderboard which consisted of 37 teams in Subtask 1A, 24 teams in Subtask 1B, and 21 teams in Subtask 1C.

5.1 Key Observations

- Ensemble methods have a systematic positive effect on single-task performance. Hard-voting ensemble strategy showed considerable improvement on both single-task subtasks. In Subtask 1A, when four models were combined, the resulting test micro-F1 was 73.05, which is a 1.06-point improvement on the highest-performing single model (XLM-RoBERTa-large: 71.99). Likewise, in Subtask 1B, the three-model ensemble scored 72.79 micro-F1, which was 0.74 points higher than the highest single model (IndicBERTv2: 72.05).

- IndicBERTv2 is an efficient multi-task learner, surpassing peers in a wide variety of goals. IndicBERTv2 was the top performer in the multi-task context of Subtask 1C with 72.40 micro-F1, which is significantly higher than the performance of XLM-RoBERTa-base(71.14) and mDeBERTa-v3-base(69.09).

- XLM-RoBERTa-large demonstrates good single-model performance. Although not explicitly trained on Indic languages, XLM-RoBERTa-large achieved competitive results as a general model, with a micro-F1 of 71.99 on Subtask 1A and 71.75 on Subtask 1B. This illustrates the high cross-lingual transfer of large-scale multilingual models.

- Transformer-based methods are far better than their traditional baselines. All the transformer models significantly surpassed the performance of n-gram baselines across the subtasks, and the usefulness of pre-trained contextual representations to detect hate speech in Bangla.

System	Micro F1 (Dev)	Micro F1 (Test)	Rank
1A: Hate Type			
XLM-R-large	73.77	71.99	—
MuRIL	73.89	71.25	—
mDeBERTa	72.77	71.58	—
IndicBERTv2	72.73	71.11	—
Ensemble above all	**75.72**	**73.05**	**7/37**
1B: Target			
IndicBERTv2	73.37	72.05	—
mDeBERTa	73.01	71.89	—
XLM-R-large	73.01	71.75	—
Ensemble above all	**74.56**	**72.79**	**8/24**
1C: Multi-task			
XLM-R-base	72.50	71.14	—
mDeBERTa	67.81	69.09	—
IndicBERTv2	**74.59**	**72.40**	**10/21**

Table 3: Performance across subtasks measured using official evaluation metrics (Micro-F1). Bold indicates final submission.

5.2 Error Analysis

Analysis of IndicBERTv2 multi-task model indicates that there are specific patterns of errors:

- **Hate Type Confusion**: Severe overlap between *Abusive* and *Political Hate* indicating the possible overlap of linguistic indicators in the Bangla social media discourse.

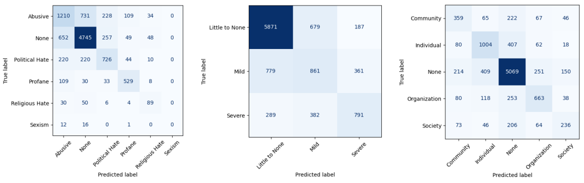
- **Severity Calibration Challenges**: Most frequent mistake (26.25%) with regular *Mild* and *Little to None* confusion which shows that intensity measurement is subjective.

- **Target Ambiguity**: Notable confusion between *Organization*, *Individual*, and *Community* targets, reflecting complex entity references in informal language.

Figure 3 provides quantitative evidence for these error patterns, with detailed analysis revealing specific model weaknesses.

Type	Text	Actual	Pred.
HT	বিদ্যুৎ জ্বালানি খাতে আওয়ামী লীগের আমলে সবচেয়ে বেশি দুর্নীতি...	Abusive	Political Hate
HT	ভারতীয় দালাল সময় টিভিকে বয়কট করুন...	Political Hate	Abusive
HT	কত সুন্দর করে বড় ভাই বলছে শাট টা কিনে নিছ ভাইয়া হালার পু হাল...	Abusive	Profane
SC	ইজরায়েলের বিচার হওয়া উচিৎ...	Mild	Little to None
SC	হেন কাপ পুলিশের মারে... বিচার হবে কি...	Little to None	Mild
SC	কিছুই হবে না বদমাশ ইসরাইল...	Severe	Mild
TA	হাসিনা তাহলে বি এন পিকে ভয় পেয়েছে...	Organization	Individual
TA	আমলিগরা হচ্ছে ছুর... তিস্তা নদির পান...	Organization	Community
TA	আমি হালায় টাকার অভাবে বিয়ে করতে পারতেছি না...	Individual	Community

Table 4: Qualitative error examples with actual vs. predicted labels. HT = Hate Type, SC = Severity Calibration, TA = Target Ambiguity.



(a) Hate Type (b) Severity (c) Target

Figure 3: Confusion matrices on the test set.

Hate Type Confusion: The most frequent confusion occurs between *Abusive* and *Political Hate* categories, with **448** total misclassifications (228 Abusive→Political, 220 Political→Abusive). Additionally, substantial non-*None* content is misclassified as *None*, including **731** Abusive, **220** Political Hate, and **30** Profane instances.

Severity Calibration: Errors concentrate at the low-moderate severity boundary, with **679** *Little-to-None→Mild* and **779** *Mild→Little-to-None* misclassifications, indicating significant threshold uncertainty.

Target Ambiguity: Notable confusion exists among *Organization*, *Individual*, and *Community* targets. Many targeted cases are incorrectly predicted as *None*, affecting **407** Individual, **253** Organization, **222** Community, and **206** Society instances.

These systematic errors highlight key challenges in fine-grained hate speech analysis for low-resource languages and outline clear priorities for future model refinement.

5.3 Cost Analysis

In real-world scenarios, particularly for low-resource languages, the computational cost and environmental impact of a model are as critical as its predictive performance. While billion-parameter models such as **mT5-Large** (Xue et al., 2021), **mT5-XL** (Xue et al., 2021), and **BLOOMZ-1B7** (Muennighoff et al., 2022) achieve strong results, their prohibitive resource demands limit accessibility and scalability. Our work instead emphasizes parameter efficiency, showing that carefully selected compact encoders can offer competitive performance at a fraction of the cost. Table 5 presents the verified parameter counts of our encoders compared to widely used large multilingual models.

Model	Parameters
mT5-Large	1.2B
mT5-XL	3.7B
BLOOMZ-1B7	1.7B
XLM-RoBERTa-large	559M
mDeBERTa-v3-base	276M
MuRIL-base-cased	177M
IndicBERTv2-MLM-only	278M

Table 5: Parameter count comparison highlighting the large gap between our compact encoders (bottom group) and billion-scale models (top group).

Our largest encoder, XLM-RoBERTa-large (559M), is under half the size of the smallest billion-scale model, while others are an order smaller (under 280M). Following scaling trends (Kaplan et al., 2020; Tay et al., 2020), these models use roughly 3–5x less GPU memory and train several times faster, offering a strong balance between accuracy, efficiency, and sustainability for multilingual hate speech detection.

6 Conclusion

The paper introduces a low-cost method to detect hate speech in Bangla with transformer ensembles and multi-task learning. Our findings indicate that by employing well-crafted ensemble techniques to multilingual transformers we can achieve competitive results relative to large individual models, at the same time being computationally efficient. The multi-task learning paradigm proved to be especially successful in capitalizing on common representations among related classification problems. We believe that an increase in the size and diversity of the dataset will result in better performance of our approach in many respects, thus expanding the range of potential uses to other text classification tasks in Bangla.

Limitations

This study is subject to several limitations. The primary challenge was the scarcity of high-quality, diverse Bangla hate speech data, which is exacerbated by the sensitive nature of the content and the limited number of publicly available sources. While the dataset exhibits class imbalance, preliminary experiments with class weighting and data sampling techniques did not yield significant performance improvements. Consequently, no explicit imbalance-handling strategy was implemented in the final models, which may adversely affect performance on minority classes. Additionally, the subjective nature of hate speech annotation, particularly for severity calibration, likely introduced labeling inconsistencies that are not fully captured by our error analysis.

References

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Angel Felipe Magnossão de Paula, Imene Bensalem, Paolo Rosso, and Wajdi Zaghouani. 2025. Transformers and ensemble methods: A solution for hate speech detection in arabic languages. *Preprint*, arXiv:2303.09823.

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

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. *Preprint*, arXiv:2103.10730.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, and 1 others. 2022. Crosslingual generalization through multitask finetuning.

Jian Ni, Taesun Moon, Parul Awasthy, and Radu Florian. 2020. Cross-lingual relation extraction with transformers. *Preprint*, arXiv:2010.08652.

Flor Miriam Plaza-Del-Arco, M. Dolores Molina-González, L. Alfonso Ureña-López, and María Teresa Martín-Valdivia. 2021. A multi-task learning approach to hate speech detection leveraging sentiment analysis. *IEEE Access*, 9:112478–112489.

Debjoy Saha, Naman Paharia, Debajit Chakraborty, Punyajoy Saha, and Animesh Mukherjee. 2021. Hate-alert@DravidianLangTech-EACL2021: Ensembling strategies for transformer-based offensive language detection. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 270–276, Kyiv. Association for Computational Linguistics.

Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2021. NLP-CUET@DravidianLangTech-EACL2021: Offensive language detection from multilingual code-mixed text using transformers. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 255–261, Kyiv. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

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.

A Experimental Setup and Hyperparameters

All fine-tuning experiments were conducted on Google Colab using a single NVIDIA T4 GPU (16 GB VRAM). To ensure reproducibility, a fixed random seed of 42 was used across all runs. Common training configurations included mixed-precision training (fp16), a maximum sequence length of 512, and the AdamW optimizer. Model-specific hyperparameters, determined through a limited search on a held-out validation set, are detailed in the following subsections.

Subtask 1A: Hate Type Classification

For this subtask, all models shared a common base configuration: they were trained for 4 epochs with a linear learning rate scheduler, a weight decay of 0.01, a warmup ratio of 0.01, and a maximum sequence length of 512. The effective batch size was standardized to 16, achieved either directly or through gradient accumulation. The distinct learning rates and gradient accumulation steps for each model are provided in Table 6.

Model	Learning Rate	BS	Grad. Ac.
XLM-RoBERTa-large	1.5×10^{-5}	4	4
mDeBERTa-v3-base	2×10^{-5}	16	1
IndicBERTv2-MLM-only	2×10^{-5}	16	1
MuRIL-base	2×10^{-5}	16	1

Table 6: Hyperparameters for Subtask 1A. Abbreviations: BS = Batch Size, Grad. Accum. = Gradient Accumulation Steps.

Subtask 1B: Target Identification

The setup for Subtask 1B was similar, employing a linear scheduler, weight decay of 0.01, and an effective batch size of 16. However, the number of epochs and warmup ratio were tuned specifically for this task. The model-specific learning rates, epochs, and warmup ratios are summarized in Table 7.

Model	LR	Epochs	WR
XLM-RoBERTa-large	1.5×10^{-5}	4	0.01
mDeBERTa-v3-base	2×10^{-5}	3	0.1
IndicBERTv2-MLM-only	2×10^{-5}	3	0.1

Table 7: Fine-tuning hyperparameters for Subtask 1B models. Column abbreviations: LR = Learning Rate, WR = Warmup Ratio.

Model	LR	Epochs	WR	Sched.
IndicBERTv2-MLM-only	2×10^{-5}	4	0.01	Cosine
XLM-RoBERTa-base	2×10^{-5}	3	—	Linear
mDeBERTa-v3-base	2×10^{-5}	4	0.1	Cosine

Table 8: Fine-tuning hyperparameters for Subtask 1C models. Column abbreviations: LR = Learning Rate, WR = Warmup Ratio, Sched. = Scheduler.

Subtask 1C: Multi-task Classification

For the multi-task learning setup (Subtask 1C), we explored different learning rate schedulers. All models were trained with an effective batch size of 16 and a fixed random seed of 42. The complete hyperparameter set for each multi-task model is detailed in Table 8.

B Detailed Performance Metrics

This appendix provides complete performance evaluations for our final models across all subtasks. We report per class precision, recall, and F1 scores to supplement the main paper's Micro F1 results.

The official evaluation metric, Micro F1, is derived from global counts across all classes. Micro Precision is calculated as $\frac{\sum TP}{\sum TP+\sum FP}$, Micro Recall as $\frac{\sum TP}{\sum TP+\sum FN}$, and Micro F1 as their harmonic mean.

We also provide Macro F1 scores, computed as the arithmetic mean of per class F1 scores, to ensure balanced performance across all categories. These metrics collectively offer a comprehensive assessment of model effectiveness for multilingual hate speech detection.

Label	Precision	Recall	F1
Abusive	57.34	53.24	55.21
Political Hate	58.76	61.31	60.01
Profane	72.76	81.38	76.83
Religious Hate	48.89	49.16	49.03
Sexism	33.33	3.45	6.25
None	82.80	83.57	83.18
Macro Avg	58.98	55.35	55.09
Micro Avg	**73.05**	**73.05**	**73.05**

Table 9: Per-class precision, recall, and F1-scores for Subtask 1A (Hate Type).

Target Type (per-class)	**Precision**	**Recall**	**F1**
Community	44.29	45.98	45.12
Individual	64.57	64.04	64.30
Organization	61.59	59.29	60.42
Society	48.64	43.04	45.67
None	82.66	84.00	83.32
Macro Avg	60.35	59.27	59.77
Micro Avg	**72.79**	**72.79**	**72.79**

Table 10: Per-class precision, recall, and F1-scores for Subtask 1B (Target Type).

Hate Type (per-class)	**Precision**	**Recall**	**F1**
Abusive	54.19	52.34	53.25
None	81.92	82.51	82.21
Political Hate	58.08	59.51	58.79
Profane	71.88	74.61	73.22
Religious Hate	47.09	49.72	48.37
Sexism	0.00	0.00	0.00
Macro Avg	52.19	53.11	52.64
Micro Avg	**71.56**	**71.56**	**71.56**
Hate Severity (per-class)			
Little to None	84.61	87.15	85.86
Mild	44.80	43.03	43.89
Severe	59.07	54.10	56.48
Macro Avg	62.83	61.43	62.08
Micro Avg	**73.75**	**73.75**	**73.75**
To Whom (per-class)			
Community	44.54	47.30	45.88
Individual	61.14	63.91	62.50
None	82.33	83.19	82.76
Organization	59.89	57.55	58.70
Society	48.36	37.76	42.41
Macro Avg	59.25	57.94	58.45
Micro Avg	**71.87**	**71.87**	**71.87**
Overall Averages (across tasks)			
Micro Avg	**72.40**	**72.40**	**72.40**
Macro Avg	58.09	57.49	57.72

Table 11: Per-class results for Subtask 1C (multi-task model) across Hate Type, Hate Severity, and Target

Heisenberg at BLP-2025 Task 1: Bangla Hate Speech Classification using Pretrained Language Models and Data Augmentation

Warning: This research paper contains examples of hateful content.

Samin Yasir
Department of Computer Science and Engineering
Shahjalal University of Science and Technology
saminyasir.cs@gmail.com

Abstract

Detecting hate speech in Bangla is challenging due to its complex vocabulary, spelling variations, and region-specific word usage. However, effective detection is essential to ensure safer social media spaces and to take appropriate action against perpetrators. In this study, we report our participation in Subtask A of Task 1: Bangla Hate Speech Detection (Hasan et al., 2025b). In addition to the provided 50K Bangla comments (Hasan et al., 2025a), we collected approximately 4K Bangla comments and employed several data augmentation techniques. We evaluated several transformer-based models (e.g., BanglaBERT, BanglaT5, BanglaHateBERT), achieving the best performance with a micro-F1 score of 71% and securing 18th place in the Evaluation Phase.

1 Introduction

As social media continues to grow in popularity, particularly among children and adolescents (Lenhart et al., 2010; Li et al., 2021), it is imperative to address hateful content. Therefore, effective detection and restriction of hate comments on the internet is necessary.

Identification of hate speech in the English language has reached an accuracy of 98.0% (Saleh et al., 2021), allowing platforms to identify most of the offensive contents and take appropriate action according to the social media platform's terms and policies (Schmidt and Wiegand, 2017). Completely banning hate speech also can be seen as a restriction of freedom of speech on the internet. If a hate comment is not protected by the moral right to freedom of expression, it falls under a moral duty to refrain from hate speech, or against the law of the state, restrictions or banning on the comment can be applied (Howard, 2019).

With over 173.8 million Bengali speakers in Bangladesh, of whom approximately 45.0 million are active social media users (Sarkar, 2024; Haque et al., 2023), the development of an effective hate speech detection system is crucial for ensuring a safer online environment for this large community. Advancing research in this domain not only safeguards users but also contributes to the broader objective of enhancing large language models (LLMs) to identify hate speech across diverse languages, thereby enabling them to issue warnings or implement preventive measures when necessary.

Detecting hate speech in Bangla contains bigger challenges due to its morphological richness with diverse synonyms (Farzana, 2021; Ali et al., 2008), regional variations, and context-based meanings. Therefore, hate speech detection models for Bangla remain less effective with a small amount of data (Tanvir Alam and Mofijul Islam, 2018). Consequently, children remain vulnerable to harmful content, while individuals spreading hate in Bangla often go undetected and unpunished. However, recent models, such as BanglaHateBERT, showed prominent performance on the hate speech detection task (Jahan et al., 2022).

In this study, Several Bangla-specific transformer models, including BanglaT5 (Bhattacharjee et al., 2023), BanglaBERT (Bhattacharjee et al., 2021), BanglaHateBERT (Jahan et al., 2022), are experimented with different types of data augmentation methods.

Our contribution can be summarized as follows:

- Experimented with five transformer-based models to achieve a micro-F1 score of 71%
- Newly collected 3,874 data-points from Bangla YouTube comments and added to the training dataset
- Analyzed the errors in the dataset to find limitations

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 476–484
December 23, 2025 ©2025 Association for Computational Linguistics

2 Related Works

Hate speech detection is a problem that researchers have been working to improve over the past few decades (Tontodimamma et al., 2021). However, it remains a challenging task for many reasons. The definition of hate speech varies significantly across regions, time periods, and different political, economic, and social contexts (Parekh, 2006).

Overfitting behavior is found to be very frequent among hate speech detection systems because the domain is vast, covering areas such as race, religion, gender, sexuality, etc in any language (Moy et al., 2021). However, a multilingual online hate speech detection system has been developed to identify hate speech in English, Italian, and German, demonstrating satisfactory performance in these languages (Corazza et al., 2020).

In Bangla, various transformer-based models have been utilized to identify offensive content from Banglish Facebook comments in a multi-label setup (Raihan et al., 2023). However, a survey on textual hate speech detection highlights that despite the advances of deep learning, particularly transformer-based models, progress is limited by weak datasets, inconsistent definitions, and poor generalization (Alkomah and Ma, 2022). Hence, these challenges are especially pronounced for Bangla, where datasets remain scarce (Romim et al., 2021).

In their work, (Hossain Junaid et al., 2021) evaluates machine learning and deep learning approaches for Bangla hate speech detection, reporting that logistic regression achieved the highest accuracy (96.2%) among machine learning methods, while a GRU-based model outperformed all approaches. In contrast, applying SVM and Naive Bayes to 1,339 Bangla samples with Naive Bayes reached a maximum accuracy of 72% (Ahammed et al., 2019). These results suggest that deep learning models are generally more effective than traditional machine learning methods for this task.

In a study on benchmarking transformer models for violence detection in Bangla YouTube comments, and showed that data augmentation with 500 samples improved F1 scores, emphasizing the value of additional context-specific data (Saha and Nanda, 2023). Another work (Sharif et al., 2022) introduced a multi-label Bangla dataset of aggressive sentences, where BanglaBERT achieved the highest weighted F1-scores (92%) in detection. These findings indicate that expanding the dataset and using BanglaBERT can increase accuracy in our task.

Similarly, the research (Romim et al., 2022) benchmarked multiple models across eight Bangla datasets by exploring various model–feature combinations and reporting variations in F1-scores. Their frequency-based word cloud analysis of traditional and non-traditional swear words informed our data augmentation, where high-frequency terms were incorporated into the training dataset.

A detailed description of the data collection strategy has been described in the research (Haider et al., 2025), which we have followed in our research. Four sequential steps are followed in the paper to generate the dataset Figure 1. Moreover, back-translating (Bangla -> English -> Bangla) dataset approach was performed to add diversity into the dataset, where the GRU and Attention techniques provided high accuracy up to 98% (Faruqe et al., 2023). Most of the research shows that the model BanglaBERT performs best for detecting Bangla hate speeches, where the data has variation (Tariquzzaman et al., 2023; Bhattacharjee et al., 2022; Das et al., 2022). Data Augmentation using translation and back-translation is discussed as an effective method to gain better accuracy in some research with the Bangla dataset (Tariquzzaman et al., 2024; Aziz and Islam, 2025; Khandaker et al., 2025).

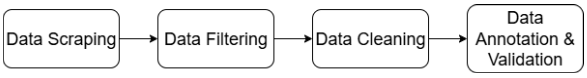


Figure 1: Data Augmentation Steps

3 System Description

This section describes how the system classifies Bangla hate content, dataset description, and different augmentation techniques used to achieve the highest micro-F1 score. All the code[1] and datasets[2] used for the task are publicly available.

3.1 Task Description

The goal of the shared task is to recognize Bangla hate comments. The input is Bangla sentences, and the output is to detect the type of hate. Both input

[1] https://github.com/Heisenberg71/blp25_task1/blob/main/example_scripts/subtask_1A_DistilBERT_example.ipynb?short_path=3504d21

[2] https://github.com/Heisenberg71/blp25_task1/tree/main/data/subtask_1A

and output is in TSV file format. There are a total of 6 types of hate classification: **Abusive, Sexism, Religious Hate, Political Hate, Profane,** and **None**.

3.2 Initial Dataset Description

The initial dataset is provided by the shared task organizers that contains about 35K labeled Bangla comments from YouTube for training (Hasan et al., 2025a). An example of a training dataset is given in Table 8. The labels are shown with frequency and percentages in the training data set, where the majority of hate types are **None** Table 1.

Label	**Frequecy**	**Percentage**
Abusive	8,212	23.12%
Sexism	122	0.34%
Religious Hate	676	1.90%
Political Hate	4,227	11.90%
Profane	2,331	6.56%
None	19,954	56.16%
Total	35,522	100%

Table 1: Training data frequency of the provided Dataset

3.3 Data Collection

We have extracted additional 3,874 Bangla comments from two videos[3] from a very popular political YouTube channel[4] in Bangladesh. We have used an online tool named[5] for collecting comments.

The collected comments contained URLs, emails, digits, punctuation marks, emojis, letters from other languages(English, Hindi, Arabic, etc), and special symbols that are unnecessary and holds little to no information on hate classification. Therefore, it was cleaned using BNLP's CleanText text cleaning package[6] and manual review. A summary of the collected data set is stated in Table 6 and Figure 2.

The collected data is used as testing data, and BanglaBERT is used to label the type of hate. Finally, the annotated data points are added to the testing dataset.

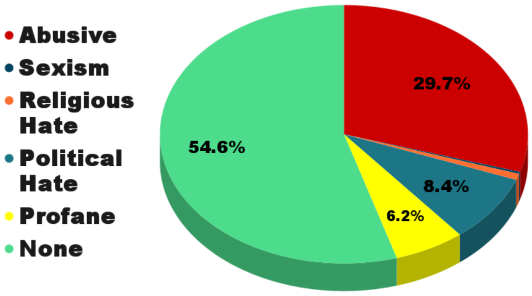


Figure 2: Distribution of hate percentage across labels

3.4 Data Augmentation: Synonym-based

Synonym-based augmentation was applied to the collected dataset to increase its variability while preserving semantic consistency. Specifically, a set of words frequently used in Bangla hate speech was identified (Romim et al., 2022), and selected words in the dataset were randomly replaced with their synonyms. This ensured that the type of hate expressed in the comments remained unchanged while introducing linguistic diversity. An illustrative example of this process is provided in Table 2.

Words	**Synonyms**
শালা	শালার পো
কুত্তা	কুত্তার বাচ্চা
খা*কি	বেশ্যা
হারামি	হারামজাদা
জা*জ	বে*ন্মা
বাল	বালছাল
জনোয়ার	পশু
দুঃখ	কষ্ট
শত্রু	প্রতিপক্ষ
দুর্গন্ধ	দুর্গন্ধময়
ক্ষমা	মাফ
দুর্গম	দুস্তর

Table 2: Some example of words that replace randomly on the dataset

3.5 Data Augmentation: Back-translation

We augmented the dataset through back-translation, translating the original Bangla data into English and then translating it back to Bangla using the Google Translate API, which introduced lexical and syntactic variations. A total of 27,000 data points were processed using this approach. While back-translation proved

[3] https://www.youtube.com/watch?v=GvDDgxbfSYk, https://www.youtube.com/watch?v=Qb0YZc1K_-8

[4] https://www.youtube.com/c/pinakibhattacharya

[5] https://youtubecommentsdownloader.com/

[6] https://github.com/sagorbrur/bnlp

effective in increasing diversity within the dataset, we observed that it frequently altered the type of hate expressed in certain comments. Such semantic shifts caused different between the original and translated labels, making the dataset unsuitable for training without extensive human review and relabeling. Given the impracticality of manually verifying a dataset of this scale, we ultimately decided not to use the dataset for model training. Illustrative examples of back-translated comments are provided in Table 9, and a detailed summary is presented in Table 7 and Figure 3.

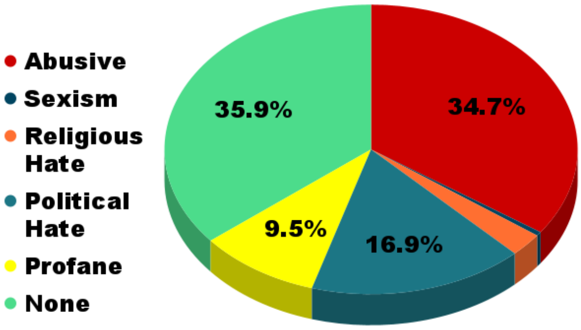


Figure 3: Distribution of hate percentage across labels in the back-translated dataset

3.6 Methodology

Approaches taken to improve the micro-F1 scores are described below.

Tokenization We used the Basic Tokenizer from BNLP[7] , which is specifically designed for Bangla text. This tokenizer provided more effective preprocessing of Bangla comments, thereby enabling the models to learn linguistic patterns more accurately and improving their ability to detect hate speech.

Stopwords Many Bangla words do not contribute significantly to the meaning of a sentence, and their removal does not alter the underlying semantics. To address this, we employed a stopword removal tool from the Bangla corpus to eliminate stopwords and punctuations. This preprocessing step ensures that the model receives only the meaningful components of a sentence as input. A list of commonly used Bangla stopwords is provided in Table 12.

Models The initial configuration was set to a single epoch. Through iterative experimentation, we found that training the model for three epochs yielded better performance within a shorter training time. Increasing the number of epochs beyond three led to overfitting, resulting in poor generalization on the test set. For model selection, we initially experimented with DistilBERT, but subsequently trained HateBERT, BanglaBERT, BanglaHateBERT, and BanglaT5 on the preprocessed dataset. Among these, BanglaBERT achieved the best performance on the provided test set. The hyperparameter settings for the experiments are summarized in Table 3.

Hyperparameters	**Details**
Dropout rate	0.1
Number of epochs	3
Training, validation, test split ratio	80:5:20
Learning rate	$2e^{-5}$
Optimizer	AdamW

Table 3: Hyperparameters of models (DistillBERT, HateBERT, BanglaBERT, BanglaT5, and BanglaHateBERT)

3.7 Results and Discussion

We experimented with different models to achieve the best micro-F1 score. Figure 4 presents the performance of these models, where BanglaBERT with the augmented dataset achieved the highest micro-F1 score of 0.71 on the released test set during the evaluation phase. Although models such as BanglaHateBERT and BanglaT5 were also used, BanglaBERT consistently outperformed them. The hyperparameters used for fine-tuning the models are provided in Table 3.

We observed that models specifically trained for the Bangla language, such as BanglaHateBERT and BanglaBERT, outperformed general hate speech detection models like HateBERT. Furthermore, data augmentation proved to be a crucial factor in our study, enhancing the performance of BanglaBERT and achieving the highest micro-F1 score of 71%.

However, due to the limited number of label samples of **Sexism and Religious Hate** in the training set, none of the models were able to identify comments belonging to this category effectively. A more detailed analysis of this limitation is provided in the error analysis section.

[7] https://github.com/sagorbrur/bnlp/tree/main

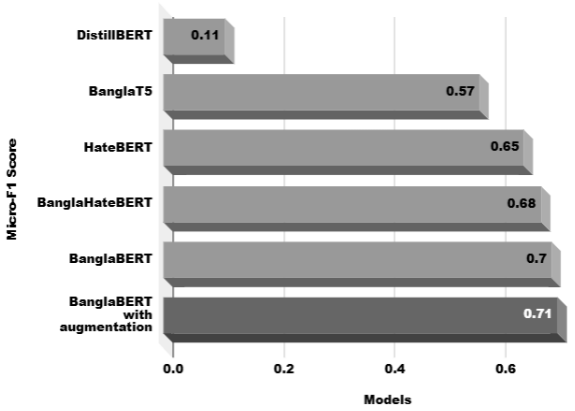


Figure 4: Micro-F1 scores of different approaches

4 Error Analysis

The representation of **Sexism** and **Religious Hate** in the dataset are extremely limited. Only 122 instances labeled as Sexism are present in the 35,523 training data points, accounting for merely 0.34% of the total training set. Consequently, every model we used fails to identify any occurrences of sexism in the test set, primarily due to the insufficient availability of training examples required to effectively learn and detect this category Table 4.

Data	# data	# Sexism	P.
Train	35,523	122	0.34%
Test	10,200	29	0.28%

Table 4: Dataset description for **Sexism**. P.: Percentage

Similarly, only 1.90% training data points is on religious hate making it difficult to detect Table 5.

Data	# data	# R.H.	P.
Train	35,523	676	1.90%
Test	10,200	179	1.75%

Table 5: Dataset description for **Religious Hate**. R.H.: Religious Hate, P.: Percentage

Some comments fall within the scope of multiple hate labels. However, assigning only one label to such comments creates accuracy issues. A single comment may correspond to two or more labels, but our model can output only one of the detected labels. If the test data assigns a different label than the one predicted, the model's score decreases, even though it has correctly identified the comment as hateful in Table 10 containing multiple hate type in a comment. Variation of hate detection between labels in the test set and BanglaBERT-generated labels.

There are also some data points on the test set that are not labeled correctly. However, BanglaBERT was able to label them correctly Table 11.

5 Conclusion and Future Scopes

This paper states the experiments we have performed to complete the shared task. Using the Bangla Tokenizer and the stopword removal technique is proven to be a very good pre-processing technique. We have collected comments from YouTube and labeled them carefully, then add them to the training set, and that improved the overall micro-F1 score. Lastly, we have utilized various well-known models that have demonstrated effectiveness in generating good results in Bangla. Among these, BanglaBERT performed best for our test and training datasets. Future studies will investigate the capabilities of LLMs and explainable hate speech detection for Bangla.

6 Limitations

The synonym-based dataset often make illogical sentences. The translators that are available are not good enough to preserve the whole meaning of a sentence. Moreover, comments are now dependent on recent political or socio-economic events. Therefore, a comment can be normal, but according to context, a normal sentence meaning can change to a hateful comment.

The model tested here is for specialized for the Bangla language. But, there are many multilingual models exists that can be very good to detect hateful comments over the internet, which have not been experimented with in this research. Similarly, the Bangla Basic tokenizer has been tried only in research. But, there are other Bangla tokenizers exists that can be improve the F1 score of the dataset.

References

Shovon Ahammed, Mostafizur Rahman, Mahedi Hasan Niloy, and S. M. Mazharul Hoque Chowdhury. 2019. Implementation of machine learning to detect hate speech in bangla language. In *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, pages 317–320.

Md. Nawab Yousuf Ali, S. M. Abdullah Al-Mamun, Jugal Krishna Das, and Abu Mohammad Nurannabi.

2008. Morphological analysis of bangla words for universal networking language. In *2008 Third International Conference on Digital Information Management*, pages 532–537.

Fatimah Alkomah and Xiaogang Ma. 2022. A literature review of textual hate speech detection methods and datasets. *Information*, 13(6).

Faisal Ibn Aziz and Muhammad Nazrul Islam. 2025. Banglahealth: A bengali paraphrase dataset on health domain. *Data in Brief*, 61:111699.

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, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *arXiv preprint arXiv:2101.00204*.

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.

Michele Corazza, Stefano Menini, Elena Cabrio, Sara Tonelli, and Serena Villata. 2020. A multilingual evaluation for online hate speech detection. *ACM Trans. Internet Technol.*, 20(2).

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in Bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Omar Faruqe, Mubassir Jahan, Md. Faisal, Md. Shahidul Islam, and Riasat Khan. 2023. Bangla hate speech detection system using transformer-based nlp and deep learning techniques. In *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–6.

Afifa Farzana. 2021. A comparative study between english and bangla: The perspective of phonemics, morphology and syntax.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Md Sakib Ul Rahman Sourove, Deeparghya Dutta Barua, Md Fahim, and Md Farhad Alam Bhuiyan. 2025. BanTH: A multi-label hate speech detection dataset for transliterated Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7217–7236, Albuquerque, New Mexico. Association for Computational Linguistics.

Rezaul Haque, Naimul Islam, Mayisha Tasneem, and Amit Kumar Das. 2023. Multi-class sentiment classification on bengali social media comments using machine learning. *International Journal of Cognitive Computing in Engineering*, 4:21–35.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Mohd. Istiaq Hossain Junaid, Faisal Hossain, and Rashedur M. Rahman. 2021. Bangla hate speech detection in videos using machine learning. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0347–0351.

Jeffrey W. Howard. 2019. Free speech and hate speech. *Annual Review of Political Science*, 22(Volume 22, 2019):93–109.

Md Saroar Jahan, Mainul Haque, Nabil Arhab, and Mourad Oussalah. 2022. BanglaHateBERT: BERT for abusive language detection in Bengali. In *Proceedings of the Second International Workshop on Resources and Techniques for User Information in Abusive Language Analysis*, pages 8–15, Marseille, France. European Language Resources Association.

Md. Arafat Alam Khandaker, Ziyan Shirin Raha, Bidyarthi Paul, and Tashreef Muhammad. 2025. Bridging dialects: Translating standard bangla to regional variants using neural models. *Preprint*, arXiv:2501.05749. Accepted in 2024 27th International Conference on Computer and Information Technology (ICCIT).

A. Lenhart, K. Purcell, A. Smith, and Kathryn Zickuhr. 2010. Social media mobile internet use among teens and young adults. *Pew Internet and American Life Project*.

Wenxin Li, Xuantong Lin, Jiani Wu, Wenhan Xue, and Junxian Zhang. 2021. Impacts social media have on young generation and older adults. In *Proceedings* of the 2021 4th International Conference on Humanities Education and Social Sciences (ICHESS 2021), pages 294–300. Atlantis Press.

Tian Xiang Moy, Mafas Raheem, and Rajasvaran Logeswaran. 2021. Hate speech detection in english and non-english languages: A review of techniques and challenges. *Technology*.

Bhikhu Parekh. 2006. Hate speech. *Public Policy Research*, 12(4):213–223.

Md Nishat Raihan, Umma Tanmoy, Anika Binte Islam, Kai North, Tharindu Ranasinghe, Antonios Anastasopoulos, and Marcos Zampieri. 2023. Offensive language identification in transliterated and code-mixed Bangla. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 1–6, Singapore. Association for Computational Linguistics.

N. Romim, M. Ahmed, H. Talukder, and M. Saiful Islam. 2021. Hate speech detection in the bengali language: A dataset and its baseline evaluation. In Mohammad Shorif Uddin and Jagdish Chand Bansal, editors, *Proceedings of International Joint Conference on Advances in Computational Intelligence*, Algorithms for Intelligent Systems, pages 457–468. Springer, Singapore.

Nauros Romim, Mosahed Ahmed, Md Saiful Islam, Arnab Sen Sharma, Hriteshwar Talukder, and Mohammad Ruhul Amin. 2022. BD-SHS: A benchmark dataset for learning to detect online Bangla hate speech in different social contexts. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5153–5162, Marseille, France. European Language Resources Association.

Saumajit Saha and Albert Nanda. 2023. BanglaNLP at BLP-2023 task 1: Benchmarking different transformer models for violence inciting text detection in Bangla. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 163–167, Singapore. Association for Computational Linguistics.

Hind Saleh, Areej Alhothali, and Kawthar Moria. 2021. Detection of hate speech using bert and hate speech word embedding with deep model. *arXiv preprint arXiv:2111.01515*.

Dr. Sukanta Sarkar. 2024. Social media as a tool of communication in bangladesh: Pattern, growth and challenges. *Pakistan Journal of Media Sciences*, 5(Issue2):54–60.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2022. M-BAD: A multilabel dataset for detecting aggressive texts and their targets. In *Proceedings of the Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situations*, pages 75–85, Dublin, Ireland. Association for Computational Linguistics.

Md Tanvir Alam and Md Mofijul Islam. 2018. Bard: Bangla article classification using a new comprehensive dataset. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5.

M. Tariquzzaman, A. N. Anam, N. Haque, M. Kabir, H. Mahmud, and M. K. Hasan. 2024. Bda: Bangla text data augmentation framework. *Preprint*, arXiv:2412.08753.

Md. Tariquzzaman, Md Wasif Kader, Audwit Anam, Naimul Haque, Mohsinul Kabir, Hasan Mahmud, and Md Kamrul Hasan. 2023. the_linguists at BLP-2023 task 1: A novel informal Bangla Fasttext embedding for violence inciting text detection. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 214–219, Singapore. Association for Computational Linguistics.

A. Tontodimamma, E. Nissi, A. Sarra, and L. Fontanella. 2021. Thirty years of research into hate speech: topics of interest and their evolution. *Scientometrics*, 126:157–179.

A Appendix

Label	**Frequecy**
Abusive	1,154
Sexism	0
Religious Hate	30
Political Hate	327
Profane	241
None	2,122
Total	3,874

Table 6: Training data frequency of the collected Dataset

Label	Frequecy
Abusive	9,366
Sexism	122
Religious Hate	706
Political Hate	4,554
Profane	2,572
None	9,680
Total	27,000

Table 7: Training data frequency after back-translation

Text	Label
অতিরিক্ত এ নিজেকে বাদুর বানাইয়া ফেলছেন রে	Abusive
অযোগ্য মহিলাদের হাতে ক্ষমতা দিয়ে দেশকে রষাতলে ফেলার আগ্রহ দেশবাসীর নাই	Sexism
অথচ এরাই ৬০ লক্ষ্য ইহুদি হত্যা করেছিলো।	Religious Hate
সরকারের নীলনকশা এখন মানুষ বুঝে গেছে মানুষ এখন আর ললিপপ খায় না।	Political Hate
হালার পাছা দিয়া উত্তর কুরিয়ার পারমাণবিক বোমা মারা হোক	Profane
আমার বাস্তবে জাওয়া হলোনা আমি ইউটিউবে দেখবো। ইট আলী	None

Table 8: Sample dataset for each of the categories

Bangla	**Translated English**	**Translated Bangla**
এমন আশ্বাস আমিও দিতে পারি	I can also give such assurance	আমি এ জাতীয় আশ্বাসও দিতে পারি
সালা বাঙালি বোকা চোদা জাতি বেশি গাল মারোছ তাই হাল তোদের	Sala Bengali fools are more cheeky	সালা বাঙালি বোকা আরও চটকদার
আরো আগে কেন নিউজ হলো না।	Why not before the news happened	খবরটি হওয়ার আগে কেন নয়
ওলে বাবালে আমাল প্যান্ট ভিজে গেছে তুমি আগে দুদু খেয়ে আসো।	Ole Babale Amal pants are wet. You eat dudu first	ওলে বাবলে অমল প্যান্ট ভেজা ।আপনি প্রথমে দুদু খান
তর গুষ্টি চুদি মা*র ছেলে টাকা পাচারকারী রাস্তায় আসিস তর বাপেদের ছাড়া	Your bunch of Chudi Ma*i's son is money smuggled on the street without Father	আপনার চুদি মা*র ছেলের গুচ্ছ আসিস তারা বাবা ছাড়া রাস্তায় অর্থ পাচার করা অর্থ

Table 9: Example of back-translation in the dataset

Text	Labels on Test set	Label by Bangla BERT
ভারতীয় দালাল সময় টিভিকে বয়কট করুন	Political Hate	Abusive
কিছুই হবে না বদমাশ ইসরাইল বা নেতায়াহুআর কিছুই হবে না	Political Hate	Abusive
সময় টিভি একটা জাউড়া মিডিয়া মিথ্যা তথ্য প্রচার করে বেড়ায়	Profane	Abusive
ইনুকে কেউ বুটদেবেনা নাচতিক সালা	Abusive	Profane
তোদের বংগবন্দু তো জাহান্নামে আছে বলেছে তো তোদের আওয়ামী কুত্তা লীগ এবার জনগণ সবাই মিলে বাঁশের লাঠি দিয়ে পিটিয়ে মেরে ফেলবো	Profane	Political Hate

Table 10: Datapoints containing multiple hate type in a comment. Variation of hate detection between labels in test set and BanglaBERT generated labels

Text	Incorrect Labels on Test set	Correctly Labeled by Bangla BERT
দাজ্জালের হাতে থাকবে তাপমাত্রা নিয়ন্ত্রণ অনেক এলাকায় গাছ পালা কাটা শুরু হয়েছে	Abusive	None
বিদ্যুৎ জ্বালানি খাতে আওয়ামী লীগের আমলে সবচেয়ে বেশি দুর্নীতি হয়েছে	Abusive	Political Hate
কোন জায়গায় বিচার পাবে মানুষ সব জায়গায় দুর্নীতি সরকার দুর্নীতি করে সরকারের প্রশাসন দুর্নীতি করে এটার জন্য খুবই দুঃখের বিষয় এটা একটা হাস্যকর কাহিনী	None	Political Hate

Table 11: Correctly labeled by BanglaBERT but incorrect labels in test set

Bangla Stopwords	Meanings
এবং	and
ও	and
যে	that
কি	what/that
কিন্তু	but
আমি	I
সে	he/she
ভালো	good
অনেক	many/much
আর	and/again
আগে	before
এখন	now
পরে	many/much
থেকে	from
এবং	and

Table 12: Some example of popular Bangla Stopwords

Retriv at BLP-2025 Task 1: A Transformer Ensemble and Multi-Task Learning Approach for Bangla Hate Speech Identification

Sourav Saha, K M Nafi Asib, and Mohammed Moshiul Hoque
Department of Computer Science and Engineering
Chittagong University of Engineering & Technology, Chittagong 4349, Bangladesh
{sahasourav1170, nafi.asib}@gmail.com; moshiul_240@cuet.ac.bd

Abstract

This paper addresses the problem of Bangla hate speech identification, a socially impactful yet linguistically challenging task. As part of the "Bangla Multi-task Hate Speech Identification" shared task at the BLP Workshop, IJCNLP–AACL 2025, our team "Retriv" participated in all three subtasks: (1A) hate type classification, (1B) target group identification, and (1C) joint detection of type, severity, and target. For subtasks 1A and 1B, we employed a soft-voting ensemble of transformer models (BanglaBERT, MuRIL, IndicBERTv2). For subtask 1C, we trained three multitask variants and aggregated their predictions through a weighted voting ensemble. Our systems achieved micro-f_1 scores of 72.75% (1A) and 72.69% (1B), and a weighted micro-f_1 score of 72.62% (1C). On the shared task leaderboard, these corresponded to 9th, 10th, and 7th positions, respectively. These results highlight the promise of transformer ensembles and weighted multitask frameworks for advancing Bangla hate speech detection in low-resource contexts. We made experimental scripts publicly available for the community.[1]

1 Introduction

With the rapid growth of social media platforms, harmful content such as hate speech and offensive language has become a pressing concern, requiring effective strategies to prevent its spread. Automated detection methods have seen substantial progress in high-resource languages, aided by large datasets and transformer-based models. However, in low-resource languages like Bangla, hate speech detection remains challenging due to limited annotated resources, dialectal variation, and frequent code-mixing. Most existing work has focused on binary classification (hate vs. non-hate) or coarse multi-class labeling, leaving fine-grained dimensions such as *type*, *severity*, and *target* underexplored. To address this gap, the BLP Workshop[2] at IJCNLP-AACL 2025 (Hasan et al., 2025b) introduced a shared task comprising three subtasks, including a multitask setup, to advance fine-grained hate speech modeling in Bangla. This paper advances current research by presenting our systems developed for the shared task. The key contributions of this work are illustrated in the following:

- Proposed efficient yet competitive ensemble methods of Bangla-capable transformer models, achieving strong performance for fine-grained hate speech classification (Subtasks 1A and 1B).
- Introduced a weighted voting ensemble within a multitask learning (MTL) framework, enabling joint prediction of hate *type*, *severity*, and *target group*, and demonstrating the viability of MTL for Bangla hate speech.
- Provided a comprehensive empirical study of deep learning and transformer-based approaches, including detailed performance comparisons and error analyses, offering insights for future research in low-resource hate speech detection.

2 Related Work

Research on Bangla hate speech detection has expanded in recent years with several new datasets and modeling approaches. Das et al. (2022) developed a corpus of Bangla and Romanized Bangla posts for hate and offensive language detection, demonstrating strong results with multilingual transformers such as XLM-R and MuRIL. Saha et al. (2023) introduced Vio-lens, a dataset of social media posts linked to communal violence. In contrast, Haider et al. (2025) proposed

[1] https://github.com/sahasourav17/Retriv-BLP25-Task-1

[2] https://blp-workshop.github.io/

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 485–491
December 23, 2025 ©2025 Association for Computational Linguistics

BanTH, a multi-label dataset for transliterated Bangla that captures multiple target categories and reflects the complexity of real-world hate speech. Hasan et al. (2025a) further introduced Bangla-MultiHate, the first multi-task Bangla hate speech dataset jointly modeling *type*, *severity*, and *target*, with extensive experiments using LLMs under zero-shot and LoRA fine-tuning. Beyond datasets, Raza and Chatrath (2024) presented HarmonyNet, an ensemble framework that improves robustness in hate speech identification, and Hossain et al. (2024) proposed a multimodal approach aligning visual and textual features for hateful content detection.

Despite these advances, most Bangla work remains focused on binary or coarse multi-class classification. Multi-task learning (MTL) has been explored to capture complementary signals in hate speech detection. For example, Awal et al. (2021) introduced AngryBERT, which jointly learns *target* and *emotion* alongside hate classification, while in the Bangla context, Saha et al. (2024) proposed MulTSeA, a multitask framework for aspect-based sentiment analysis. These studies suggest that MTL is well-suited for complex tasks like hate speech, where dimensions such as type, severity, and target are interdependent. This work extends Bangla hate-speech identification to a fine-grained, multidimensional setting. Unlike prior studies focused on binary or coarse classification, we employ transformer-based ensembles and a multitask framework to jointly model *type*, *severity*, and *target group*, addressing a key gap in Bangla hate speech research.

3 Task and Dataset Descriptions

The primary aim of this shared task (Hasan et al., 2025b) is to conduct fine-grained hate speech identification in Bangla social media content, moving beyond binary detection toward multi-dimensional classification. The task was organized into multiple related subtasks:

- **Subtask 1A (Hate Type)**: Classify a text as *Abusive*, *Sexism*, *Religious Hate*, *Political Hate*, *Profane*, or *None*.
- **Subtask 1B (Target Group)**: Identify whether the hate is targeted at *Individuals*, *Organizations*, *Communities*, or *Society*.
- **Subtask 1C (Multitask)**: Jointly predict the *hate type*, *severity* (*Little to None*, *Mild*, *Severe*), and *target group*.

All three subtasks use the same dataset splits: 35,522 samples for training, 2,512 for development, and 10,200 for testing. On average, the texts contain about 78 tokens across all splits. While the split sizes are identical, the label distributions differ across subtasks. Table 1 presents the detailed label-wise distributions across train, development, and test sets for all subtasks.

Subtask	Label	Train	Dev	Test
1A (Type)	None	19,954	1,447	5,751
	Abusive	8,212	549	2,312
	Political Hate	4,227	283	1,220
	Profane	2,331	185	709
	Religious Hate	676	40	179
	Sexism	122	8	29
1B (Target)	None	21,190	1,528	6,093
	Individual	5,646	391	1,571
	Organization	3,846	292	1,152
	Community	2,635	159	759
	Society	2,205	142	625
1C (Type)	None	19,954	1,447	5,751
	Abusive	8,212	549	2,312
	Political Hate	4,227	283	1,220
	Profane	2,331	185	709
	Religious Hate	676	40	179
	Sexism	122	8	29
1C (Severity)	Little to None	23,489	1,714	6,737
	Mild	6,853	426	2,001
	Severe	5,180	372	1,462
1C (Target)	None	21,190	1,528	6,093
	Individual	5,646	391	1,571
	Organization	3,846	292	1,152
	Community	2,635	159	759
	Society	2,205	142	625

Table 1: Label distributions across Train, Dev, and Test splits for all subtasks.

These statistics highlight the inherent class imbalance, such as the small number of *Sexism* and *Religious Hate* instances in Subtask 1A, which makes the task more challenging.

4 System Description

The proposed approach leverages different architectures tailored to the specific characteristics of each subtask. For subtasks 1A and 1B, we employ a soft ensemble of three pre-trained transformer models, while for subtask 1C, we adopt a multi-task learning framework.

4.1 Text Preprocessing

We apply minimal preprocessing to preserve the authentic nature of social media content. The preprocessing pipeline consists of: (1) removal of

Bangla digits, and (2) standard tokenization using each model's respective tokenizer. We observe that the provided dataset appears well curated, requiring minimal additional cleaning. All input sequences are truncated or padded to a maximum length of 128 tokens.

4.2 Baseline Models

We utilize three complementary pre-trained transformer models across all subtasks:

- **BanglaBERT** (`csebuetnlp/banglabert`): A monolingual BERT model specifically pre-trained on Bangla text, providing strong language-specific representations. (Bhattacharjee et al., 2022)
- **MuRIL** (`google/muril-base-cased`): A multilingual model covering 17 Indian languages, including Bangla, offering cross-lingual contextual understanding. (Khanuja et al., 2021)
- **IndicBERTv2** (`ai4bharat/IndicBERTv2-MLM-only`): A model trained on 12 major Indian languages with enhanced tokenization for Indic scripts. (Doddapaneni et al., 2023)

4.3 Task-Specific Architectures

Soft Voting Ensemble Approach: For hate-type classification (1A) and target-group identification (1B), each base model is fine-tuned independently with task-specific classification heads. The final prediction is obtained through soft voting by averaging the prediction probabilities from all three models:

$$P_{\text{ensemble}}(y|x) = \frac{1}{3}\sum_{i=1}^{3} P_i(y|x) \tag{1}$$

where $P_i(y|x)$ represents the probability distribution from model i for input x.

Weighted Voting Multitask Approach For the joint prediction task, each base model is fine-tuned independently as a multitask learner with three classification heads: hate type (6 classes), severity (3 classes), and target group (4 classes). The individual multitask objective function combines cross-entropy losses from all tasks:

$$\mathcal{L}_{\text{mtl}} = \alpha\mathcal{L}_{\text{type}} + \beta\mathcal{L}_{\text{severity}} + \gamma\mathcal{L}_{\text{target}} \tag{2}$$

The final ensemble prediction uses weighted voting based on individual development set performance:

$$\begin{aligned} P_{\text{final}}(y \mid x) = 0.5\, &P_{\text{MuRIL}}(y \mid x) \\ &+ 0.3\, P_{\text{BanglaBERT}}(y \mid x) \\ &+ 0.2\, P_{\text{IndicBERTv2}}(y \mid x) \end{aligned} \tag{3}$$

where the weights reflect each model's individual performance ranking on the development set.

4.4 Training Configuration

All models are trained using the `AdamW` optimizer with identical hyperparameters across all subtasks: learning rate of 2×10^{-5}, batch size of 16, and 3 training epochs. For subtasks 1A and 1B, each model in the soft voting ensemble is trained independently with task-specific objectives. For subtask 1C, each model is trained independently as a multitask learner before combining predictions through weighted voting. The training configuration used in our experiments is summarized in Table 2.

Parameter	Value
Optimizer	AdamW
Learning rate	2×10^{-5}
Batch size	16
Epochs	3
Max sequence length	128

Table 2: Training hyperparameters used across all transformer models.

5 Results and Analysis

5.1 Performance Against Baselines

Table 3 reports system performance across all three subtasks, evaluated with the official metrics (micro-f_1 for 1A and 1B, and weighted micro-f_1 for 1C). The organizers also released three baselines-Random, Majority, and n-gram, which obtained 16.38, 56.38, and 60.20% on 1A; 20.43, 59.74, and 62.09% on 1B; and 23.04, 60.72, and 63.05% on 1C, respectively.

Our systems substantially outperform these baselines. For example, BanglaBERT attains 71.00% on subtask 1A, more than 10 points higher than the n-gram baseline. The best-performing systems are ensembles: soft voting achieves 75.72% (1A) and 74.96% (1B), while weighted voting performs best on 1C (75.12%).

Model	Micro-f_1		W-Micro-f_1
	1A	**1B**	**1C**
BiLSTM (GV)	69.39	64.49	–
BiLSTM (FT)	68.67	62.74	–
BiGRU (GV)	69.35	68.75	–
BiGRU (FT)	66.92	68.75	–
MRL	74.00	74.60	74.79
BNB	71.00	73.61	73.35
INB	74.00	73.17	71.22
SV (MRL+BNB+INB)	**75.72**	**74.96**	74.08
HV (MRL+BNB+INB)	72.53	74.16	73.42
WV (MRL+BNB+INB)	74.16	74.56	**75.12**

Table 3: Performance of employed models across all subtasks. A dash (–) denotes that the model was not evaluated for the corresponding subtask. Abbreviations: GV=GloVe, FT=FastText, MRL=MuRIL, BNB=BanglaBERT, INB=IndicBERTv2, SV=Soft Voting, HV=Hard Voting, WV=Weighted Voting.

5.2 RNNs vs. Transformers vs. Ensembles

On subtasks 1A and 1B, BiLSTM and BiGRU models with static embeddings (GloVe, FastText) yield scores in the mid-60s, significantly lower than transformer-based models. Based on these results, we did not extend RNNs to Subtask 1C.

Among transformers, MuRIL and IndicBERTv2 perform comparably and generally surpass BanglaBERT, reflecting their larger multilingual training. However, ensemble methods consistently outperform individual models. Soft voting is most effective in 1A and 1B, whereas weighted voting excels in 1C, suggesting that the optimal ensemble strategy depends on task complexity.

5.3 Error Analysis

Confusion Patterns. In subtask 1A (hate type), the system frequently confuses *Abusive* with *None*, and *Political Hate* with *Profane*. Minority categories such as *Sexism* and *Religious Hate* suffer from very low recall due to class imbalance. In subtask 1B (target group), *Organization* is often predicted as *None*, and *Society* is confused with *Individual*. Subtask 1C inherits these trends, with additional difficulty distinguishing between *Mild* and *Severe* hate severity. Confusion matrices and error examples are provided in Appendix A.

Qualitative Errors. Representative examples highlight typical misclassifications. For instance,

- **Implicit or sarcastic hate:** ভাইয়া আপনি অভিনেতা হইয়েন না না হলে সবাই বাচ্চা চাইবে *(Brother, dont act like an actor, otherwise everyone will demand children from you)* was annotated as *Abusive, Individual*, but all systems predicted *None*.

- **Subtask complementarity:** এটা রে আওয়ামী লীগ ভোট দিবে কেনমাথায় আসে না *(Why would anyone vote for Awami League, I cannot imagine)* was misclassified in Subtask 1B (*Organization*) but correctly resolved in the multitask model.

- **Trade-offs in multitask learning:** হারামি মোসাদ্দেক দেখি এইখানে *(That bastard Mosaddek, I see him here)* was correctly identified as *Profane, Individual* in single-task models, but misclassified as *Abusive, Individual* in multitask.

Effect of Label Imbalance. The dataset distribution depicted in Table 1 reveals severe class imbalance across subtasks. In Subtask 1A, categories such as *Sexism* (only 122 training instances) and *Religious Hate* (676 instances) are underrepresented, explaining their very low recall in our experiments. Similarly, in Subtask 1B, minority classes like *Community* and *Society* are frequently misclassified as *None* or *Individual*. For Subtask 1C, the dominance of the *Little to None* category in severity prediction makes it challenging for models to correctly identify *Mild* and *Severe* hate. These imbalances highlight the need for data augmentation and re-weighting strategies in future work.

5.4 Summary of Findings

Our analysis shows that (i) transformer ensembles consistently outperform single models and RNN baselines, (ii) multitask learning captures complementary signals across hate type, severity, and target group, though sometimes introducing inconsistencies, and (iii) errors stem primarily from subtle linguistic cues, overlapping class boundaries, and severe class imbalance. Together, these findings validate the complementary strengths of ensemble and multitask strategies for Bangla hate speech identification.

Official Shared Task Results. On the blind test set used for leaderboard evaluation, our submissions achieved 72.75% Micro-f1 in Subtask 1A (9th), 72.69% in Subtask 1B (10th), and 72.62% Weighted Micro-f1 in Subtask 1C (7th). These results confirm the competitiveness of our ensemble and multitask strategies in a challenging shared-task setting.

6 Conclusion

In this paper, we presented our systems for the BLP 2025 Shared Task 1 on Bangla hate speech identification. We explored transformer-based ensembles for subtasks 1A and 1B, and designed an efficient yet competitive multitask learning framework for subtask 1C. Our models achieved competitive performance across all subtasks, demonstrating the viability of both ensemble strategies and multitask learning in a low-resource setting like Bangla. The analysis further revealed challenges such as class imbalance and the difficulty of modeling underrepresented categories (e.g., *Sexism*, *Religious Hate*). For future work, we aim to explore data augmentation, cross-lingual transfer, and more robust multitask architectures to improve fine-grained hate speech detection in Bangla and extend these approaches to other low-resource languages.

7 Limitations

While our system demonstrates competitive performance on fine-grained hate speech detection, we acknowledge certain limitations. Our ensemble approaches require training and inference across multiple transformer models, increasing computational overhead compared to single-model solutions. The fixed weighting strategy for subtask 1C, while empirically determined from development performance, may benefit from more sophisticated dynamic weighting mechanisms. Additionally, our evaluation focuses on the shared task dataset, and broader cross-domain validation would strengthen the generalizability claims of our ensemble strategies.

References

Md Rabiul Awal, Rui Cao, Roy Ka-Wei Lee, and Sandra Mitrović. 2021. Angrybert: Joint learning target and emotion for hate speech detection. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 701–713. Springer.

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.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in Bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Md Sakib Ul Rahman Sourove, Deeparghya Dutta Barua, Md Fahim, and Md Farhad Alam Bhuiyan. 2025. BanTH: A multi-label hate speech detection dataset for transliterated Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7217–7236, Albuquerque, New Mexico. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Eftekhar Hossain, Omar Sharif, Mohammed Moshiul Hoque, and Sarah Masud Preum. 2024. Align before attend: Aligning visual and textual features for multimodal hateful content detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 162–174.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, and 1 others. 2021. Muril: Multilingual representations for indian languages. *arXiv preprint arXiv:2103.10730*.

Shaina Raza and Veronica Chatrath. 2024. Harmonynet: navigating hate speech detection. *Natural Language Processing Journal*, 8:100098.

Sourav Saha, Shawly Ahsan, and Mohammed Moshiul Hoque. 2024. Multsea: Aspect-based sentiment analysis using multitask learning from bengali texts. In *2024 27th International Conference on Computer and Information Technology (ICCIT)*, pages 25–31. IEEE.

Sourav Saha, Jahedul Alam Junaed, Maryam Saleki, Arnab Sen Sharma, Mohammad Rashidujjaman Rifat, Mohamed Rahouti, Syed Ishtiaque Ahmed, Nabeel Mohammed, and Mohammad Ruhul Amin. 2023. Vio-lens: A novel dataset of annotated social network posts leading to different forms of communal violence and its evaluation. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 72–84.

A Detailed Error Analysis

We include confusion matrices for all three subtasks to illustrate misclassification patterns, along with representative failure cases that highlight challenges such as class imbalance, and subtle contextual cues.

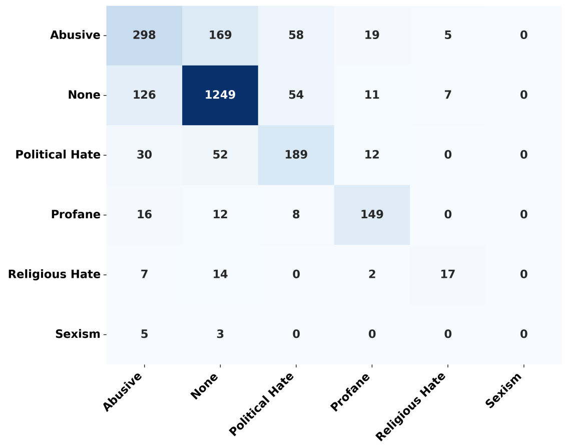


Figure 1: Confusion Matrix for Subtask 1A

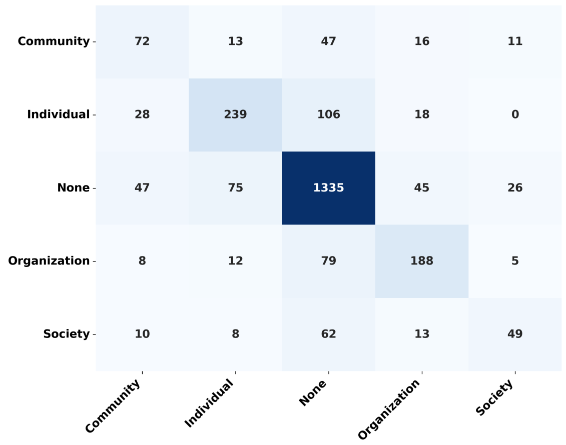
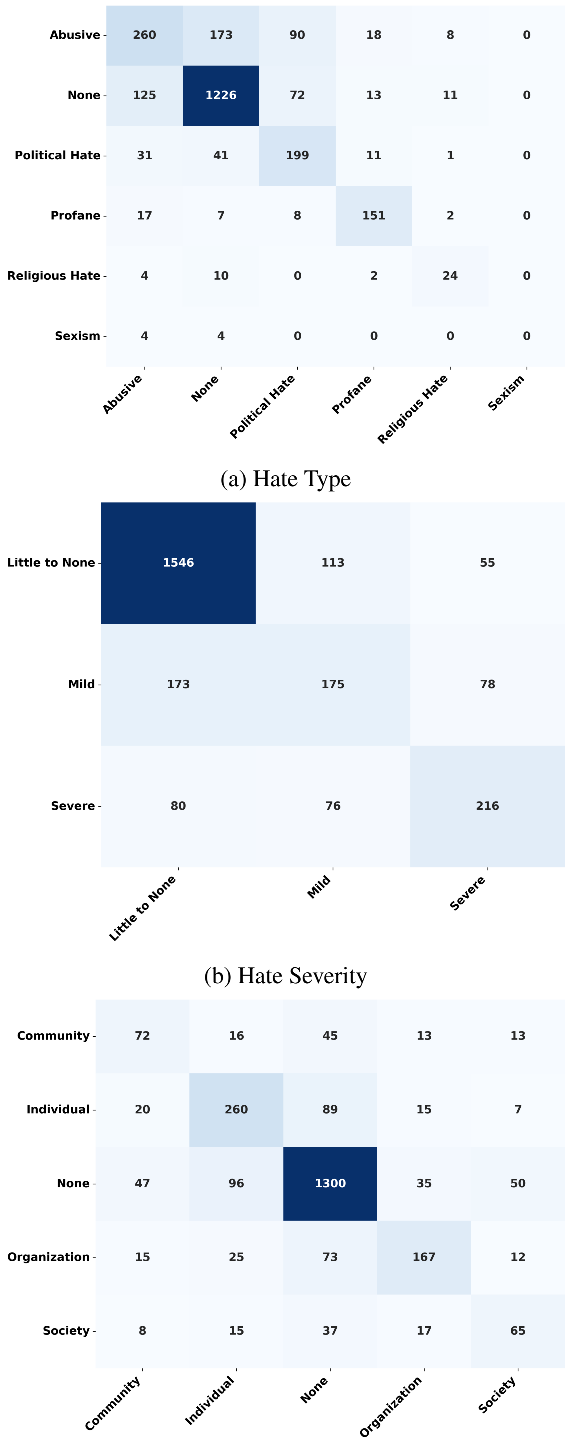


Figure 2: Confusion Matrix for Subtask 1B



(a) Hate Type

(b) Hate Severity

(c) To Whom

Figure 3: Confusion matrices for subtask 1C

Since all three subtasks were derived from the same dataset, we analyze errors using a shared set of representative examples. This allows us to highlight systematic issues across subtasks in a more consistent manner. Instead of comparing systems directly against each other, we focus on cases where each subtask fails, succeeds, or faces systematic challenges. Wrong predictions are highlighted in blue.

A.1 Subtask 1A: Hate Type Classification

Table 4 shows cases where Subtask 1A (type classification) fails. We observe frequent confusion between *Abusive* and *Profane*, as well as under-prediction of subtle *Political Hate*. These errors often arise from short texts or figurative language.

Text	**Gold Annotation**	**1A Prediction**
সয়তানর খালাম্মা কয়কি *(What did the devils aunt say?)*	Abusive	None
নির্বাচক মণ্ডলীর দের কে খেলানো দরকার হালারা *(The election committee members should be made to play, idiots.)*	Political Hate	Profane

Table 4: Examples where Subtask 1A (type) failed. Wrong predictions are in blue.

A.2 Subtask 1B: Target Group Classification

Table 5 presents errors from Subtask 1B (target group identification). The main challenge lies in distinguishing *Organization* vs. *Community*, and in cases where hate is implied but the target is indirect.

Text	**Gold Annotation**	**1B Prediction**
তোমাদের রাজনীতি হাস্যকর *(Your politics is ridiculous.)*	Organization	Community
ভালো থাকলে কাপড় খুলে বের হস ... জাহান্নামে গেলি *(If you're fine, strip off your clothes ... went to hell.)*	Community	Individual

Table 5: Examples where Subtask 1B (target) failed. Wrong predictions are in blue.

A.3 Subtask 1C: Multitask Classification

Table 6 highlights errors in Subtask 1C (multitask). Although multitask modeling captures interdependencies between type, severity, and target, we observe systematic errors such as over-predicting *Severe*, mismatched targets, and type drift.

Text	**Gold Annotation**	**1C Prediction**
গোয়া মারা দিয়ে আছে বাংলাদেশ মাদারচোদ নিউজ করে সালার পুত পাম দেস *(Bangladesh is ruined motherfucker making news and buttering them up.)*	Profane / Mild / Organization	Profane / Severe / Individual
হারামি মোসাদ্দেক দেখি এইখানে *(That bastard Mosaddek, I see him here.)*	Profane / Little to None / Individual	Abusive / Severe / Individual

Table 6: Examples where Subtask 1C (multitask) failed. Wrong predictions are in blue.

A.4 Cases Where All Subtasks Fail

Finally, Table 7 shows difficult examples where all systems fail. These include sarcasm, implicit hate, or ambiguous targets, which remain challenging for current transformer models.

Text	**Gold Annotation**	**System Predictions**
দনের নিবাচন ফাইজলামি। *(Todays election is a farce.)*	Profane / Mild / None	None / None / None
তোমরা শুধু প্রতিবাদে জানাতে পারবে ... জনগণের টাকা দিয়ে খুজতেছে জনগণকে রক্ষা করার জন্য *(You will only be able to protest ... wasting peoples money pretending to protect them.)*	Abusive / Little to None / Community	Political Hate / None / None

Table 7: Challenging examples where all subtasks failed. Wrong predictions are in blue.

B Reproducibility Note

All experiments were conducted on a single NVIDIA RTX 3090 GPU with 24 GB VRAM. We used the PyTorch[3] deep learning framework together with the HuggingFace[4] Transformers library. All hyperparameters are listed in Table 2. Random seeds were fixed across runs for consistency, although minor variations in results may occur due to non-deterministic GPU operations.

[3] https://pytorch.org/

[4] https://huggingface.co/

CoU-CU-DSG at BLP-2025 Task 1: Leveraging Weighted Probabilistic Fusion of Language Models for Bangla Hate Speech Detection

Ashraful Alam[1], **Abdul Aziz**[2], **and Abu Nowshed Chy**[3]
Department of Information and Communication Technology
[1]Comilla University, Cumilla-3506, Bangladesh
Department of Computer Science and Engineering
[2]International Islamic University Chittagong, Chattogram-4318, Bangladesh
[3]University of Chittagong, Chattogram-4331, Bangladesh
ashrafulalam.cou.ict@gmail.com, aziz.abdul.cu@gmail.com, nowshed@cu.ac.bd

Abstract

The upsurge of social media and open source platforms has created new avenues for the rapid, global spread of negativity and obscenities targeting individuals and organizations. The process to identify hate speech is critical for the lexical and regional variation as well as the morphological complexity of the texts, especially in low-resource languages, e.g. Bangla. This paper presents our participation in the Hate Speech Detection task at the second workshop on Bangla Language Processing. The objective of this task is not only to detect whether the content is hateful, but also to identify the type of hate, the target group, and its severity. We proposed a Transformer-based weighted probabilistic fusion model to detect the presence of hate speech in Bangla texts. We independently fine-tuned three pre-trained Transformer models, BanglaBERT, XLM-RoBERTa, and MuRIL, to capture diverse linguistic representations. The probability distributions obtained from each model were combined using a weighted fusion strategy, allowing the system to leverage the strengths of all models simultaneously. This fused representation was then used to predict the final labels for the given instances. The experimental results showed that our proposed method obtained competitive performance, ranking 10th in subtask 1A and 15th in subtask 1B among the participants.

1 Introduction

The rapid growth of social networks and online platforms has facilitated communication and information sharing on an unprecedented scale. However, this has also led to the proliferation of harmful content, including hate speech, which can incite violence, discrimination, and social unrest (Roy et al., 2022; Mahajan et al., 2024). Online abuse and the spread of negativity are common practices and an important social problem that is highly correlated with the emergence of social media platforms (Antypas and Camacho-Collados, 2023). Detecting hate speech in Bangla is especially challenging due to the informal language, spelling variations, and the use of slang in comment sections. While most existing hate speech detection systems have been developed for English or other high-resource languages (Toraman et al., 2022; Nozza, 2021), research in Bangla hate speech detection, particularly in YouTube comments, remains limited.

To address this gap, we participated in both Subtask 1A and Subtask 1B of the shared task (Hasan et al., 2025b). Subtask 1A and Subtask 1B are multiclass text classification problems in which each comment is categorized into one of the hate speech classes or labeled as None. We propose a Transformer-based fusion model where we fine-tuned XLM-RoBERTa, BanglaBERT (Bhattacharjee et al., 2022), and MuRIL (Khanuja et al.), under various hyperparameter settings. To handle the strong class imbalance inherent in the data, we integrate a weighted loss function during training and evaluate performance using the micro f1 metric. Our approach achieves competitive performance across both subtasks, demonstrating its effectiveness in identifying harmful Bangla content. These findings highlight the potential of Transformer-based fusion models for low-resource languages and contribute to safer online interactions for Bangla-speaking users.

2 Related Work

Hate speech detection is a growing area in research, particularly with the uprising of social media (Toraman et al., 2022; Salles et al., 2025). While existing work has been done in well-resourced languages like English (Lee et al., 2022), there remains a substantial gap in research for low-resource languages, especially those with complex linguistic structures and script adaptation challenges (Nozza, 2021), such as Bangla.

Early foundational work in hate speech detection

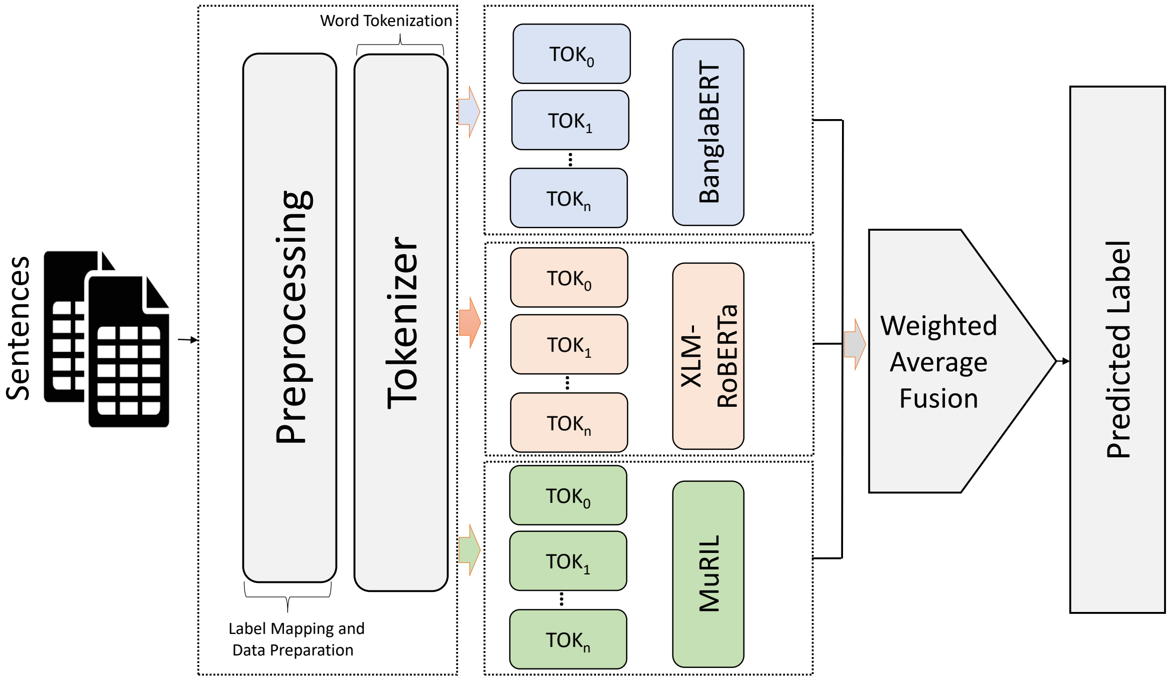


Figure 1: Our proposed model for hate speech detection.

on social media platforms like Twitter has been seen in (Talat and Hovy, 2016). Their research primarily focused on analyzing the impact of extralinguistic features, such as gender. They aimed to enhance the detection of hate speech (Singh and Thakur, 2024; Lee et al., 2022).

In the context of Bangla text, deep learning methods such as recurrent neural networks and long-short-term memory are used to handle hate speech detection in multilabel texts (Das et al., 2022). A key contribution was made by (Bhattacharjee et al., 2022), who developed an annotated dataset of 10,000 Bangla tweets, including both actual Bangla and Romanized Bangla. They implemented models like MuRIL and got excellent performance. More recently, a significant advancement in multiclass classification was made by (Bhattacharjee et al., 2022). They fine-tuned BanglaBERT on their training data. In contrast, we propose a fusion model leveraging three Transformer-based models, including BanglaBERT, XLM-RoBERTa, and MuRIL, to exploit the diverse contextual dimension of Bengali hate speech.

3 Methodology

In this section, we describe our proposed approach for the hate speech detection task. The overview of our framework is depicted in Figure 1.

Given an input text, we first map and assign weights to the labels to address class imbalance, then we employ three transformer models, including BanglaBERT (Bhattacharjee et al., 2022), XLM-RoBERTa (Antypas and Camacho-Collados, 2023), and MuRIL, to detect hate speech. Finally, for the effective fusion of the scores, we take the weighted arithmetic mean of the prediction scores of these models.

3.1 Transformer Models

Transformer models are adept at capturing long-term dependencies by leveraging multi-head attention and positioned embedding mechanisms. This approach facilitates a robust understanding of the relationships between words, which is essential for obtaining a richer, contextualized representation of the argument's context (Aziz et al., 2023). In this study on hate speech detection, we select three state-of-the-art multilingual and language-specific Transformer models as our foundation (Kim et al., 2022), including BanglaBERT, XLM-RoBERTa, and MuRIL. These models were chosen to effectively handle the lexical diversity present in our dataset, serving as the base architectures upon which our fine-tuning, Transformer-based approach is applied.

3.2 Fusion of Transformer Models

In the field of natural language processing (NLP), combining the strengths of multiple models is a standard technique to boost performance beyond the capability of any single architecture and mitigate individual model limitations. Our proposed framework adopts a fusion strategy to synthesize the capabilities of BanglaBERT, XLM-RoBERTa, and MuRIL (Romim et al., 2022).

We achieve this synergy by estimating a single, unified probability score for each classification class. This score is derived from fusing the prediction scores generated independently by each of the three fine-tuned Transformer models. Specifically, we employ the weighted probabilistic mean of these three probability scores for the fusion process. By applying weights, we can prioritize the output of the model that shows the highest reliability or relevance for a given prediction. The final label for the input text is then determined by selecting the class associated with the resulting highest fused probability score.

4 Experiments and Results

4.1 Dataset and Preprocessing

We used the official datasets (Hasan et al., 2025a) provided by the Bangla Multi-task Hate Speech Identification Shared Tasks organizers (Hasan et al., 2025b). As shown in Table 1, the training, development, and test sets contain Bangla text instances annotated with six categories for subtask 1A. On the other hand, Table 2 shows the data distribution of subtask 1B that contains 5 categories of labels, such as None, Individual, Society, Community, and Organization.

Labels	Counts
None	19954
Profane	2331
Abusive	8212
Sexism	122
Political Hate	4227
Religious Hate	676

Table 1: Label counts of training dataset (subtask 1A).

The dataset used in this study exhibited a significant class imbalance, where some classes were underrepresented compared to others. To mitigate this issue during training, we employed a weighted cross-entropy loss (Vázquez-Osorio et al., 2024).

$$\mathcal{L}_{\text{weighted}} = -\sum_{i=1}^{C} w_i \, y_i \, \log \hat{y}_i$$

Labels	Counts
None	21190
Individual	5646
Society	2205
Community	2635
Organization	3846

Table 2: Label counts of training dataset (subtask 1B).

In this formulation, C denotes the total number of classes in the dataset. The term y_i represents the ground-truth label for class i expressed in a one-hot encoded format, while $\hat{y}_i$ corresponds to the predicted probability assigned by the model to class i. The coefficient w_i is a class-specific weight that determines the relative importance of each class in the loss calculation, assigning larger penalties to misclassifications from minority classes and smaller penalties to the majority classes. The class weights w_i were derived from the distribution of the training data using the following expression:

$$w_i = \frac{N}{C \, n_i} \tag{1}$$

Here, N is the total number of training samples, n_i denotes the number of samples belonging to class i, and C again is the total number of classes. This formulation ensures that classes with fewer instances are assigned proportionally higher weights, thereby balancing the contribution of each class to the overall loss and reducing the bias towards majority classes. (Al Maruf et al., 2024). We tokenized texts separately for each model using their respective models' tokenizers, BanglaBERT, XLM-RoBERTa, and MuRIL, splitting into subword units, padding or truncating to 256 tokens, and converting them to PyTorch tensors for training, validation, and testing.

4.2 Experimental Settings

We now present the details of the experimental setup, including the specific hyperparameter configurations and fine-tuning strategies utilized to develop our proposed system. We performed the fine-tuning process using the following key configurations, which were defined within the Training Arguments class. The models were trained for three epochs. We utilized a training batch size of 16, and set the learning rate to the standard pre-trained optimization value of 2e-5. Additionally, a weight decay of 0.01 was applied to mitigate overfitting. Training logs were recorded every 50 steps. To optimize resources, we disabled immediate evaluation during training and set the model saving frequency to zero. In our weighted probabilistic fusion, we consider weights of 0.5, 0.3, and 0.2 for BanglaBERT, XLM-RoBERTa, and MuRIL, respectively, based on their individual performance.

4.3 Results and Analysis

To evaluate the performance of the participant's system at the BLP25 hate speech detection shared task (Hasan et al., 2025b), the micro f1 score is considered as the main evaluation matrics for subtask 1A and subtask 1B.

Team	Position	Score
shifat_islam	1st	0.7362
SyntaxMind	2nd	0.7345
nahidhasan	7th	0.7305
CoU-CU-DSG	10th	0.7273
pritampal98	19th	0.7057
programophile	21st	0.7013
intfloat	33rd	0.6634

Table 3: Comparative results with other selected participants (Subtask 1A).

Team	Position	Score
mahim_ju	1st	0.7356
shifat_islam	2nd	0.7335
nahidhasan	8th	0.7279
CoU-CU-DSG	15th	0.7114
pritampal98	19th	0.6974
lamiaa	24th	0.2848

Table 4: Comparative results with other selected participants (Subtask 1B).

The performance of our proposed system in the BLP25 hate speech detection shared task is analyzed across Subtask 1A and Subtask 1B in this section. Table 3 and Table 4 present the comparative results for Subtask 1A and Subtask 1B, respectively, contrasting our system with other top entries. At first, we presented the performance of our proposed system. We also presented the performance of top-ranked participating systems and the baseline used in subtask 1A and subtask 1B. Here, we see that our proposed method obtained a good score in terms of the primary evaluation metric micro f1 score.

Method	Dev set	Test set
BanglaBERT	0.7356	0.7156
XLM-R	0.6937	0.6735
MuRIL	0.6846	0.6628
BanglaBERT+XLM-R	0.7308	0.7242
BanglaBERT+MuRIL	0.7348	0.7211
XLM-R+MuRIL	0.7225	0.7078
Proposed Fusion Model	0.7456	0.7273

Table 5: Ablation study of our proposed model (Subtask 1A). XLM-R represents the XLM-RoBERTa model.

In our proposed system, we perform the effective fusion of three Transformer models. However, to validate the performance of our fusion strategy, we conduct evaluate the performance of each model used in our proposed system. The results of the ablation study of our proposed model are articulated in Table 5. From the results, it is observed that BanglaBERT performed better compared to other models when considering individual model performances. However, combining the three models' prediction scores by using a weighted average improved the performance. It shows that the fusion strategy improve the ∼2% performance compared to the BanglaBERT model and improves the ∼6% performance compared to the other two models in terms of the evaluation measure micro f1 score. This validates the importance of our fusion strategy.

5 Conclusion and Future Directions

In this paper, we present an approach to detect hate speech from Bangla texts leveraging three BERT variants, including BanglaBERT, XLM-RoBERTa, and MuRIL, with an effective weighted fusion strategy. Experimental results demonstrate the efficiency of our fusion model, which helped us to obtain a competitive position in both subtask 1A and subtask 1B.

In the future, we intend to explore feature engineering, training strategies, and other pretrained models for further improvement. We also have a plan to explore the external knowledge of other similar domains, as well as the strength of large language models (LLMs) in this task.

Limitations

Although our proposed weighted probabilistic fusion approach demonstrates promising improvements for Bangla hate speech detection, some limitations remain. First, the performance of the fused models is still bounded by the representational capacity and pretraining data of the individual language models. For example, BanglaBERT is trained primarily on curated Bangla corpora, while XLM-RoBERTa and MuRIL include multilingual data, which may introduce noise or under-represent Bangla-specific linguistic phenomena such as code-mixing, dialectal variations, or colloquial expressions. Second, despite our fusion strategy improving the performance, more adaptive or robust fusion mechanisms could potentially yield stronger results. Finally, although our method reduces variance compared to relying on a single model, it increases computational cost during inference by requiring predictions from multiple language models. This may limit practical deployment in low-resource or real-time settings.

References

Abdullah Al Maruf, Ahmad Jainul Abidin, Md Mahmudul Haque, Zakaria Masud Jiyad, Aditi Golder, Raaid Alubady, and Zeyar Aung. 2024. Hate speech detection in the bengali language: a comprehensive survey. *Journal of Big Data*, 11(1):97.

Dimosthenis Antypas and Jose Camacho-Collados. 2023. Robust hate speech detection in social media: A cross-dataset empirical evaluation. In *The 7th Workshop on Online Abuse and Harms (WOAH)*, pages 231–242, Toronto, Canada. Association for Computational Linguistics.

Abdul Aziz, Md Akram Hossain, and Abu Nowshed Chy. 2023. Csecu-dsg at semeval-2023 task 4: Fine-tuning deberta transformer model with cross-fold training and multi-sample dropout for human values identification. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 1988–1994.

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.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in bengali. 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 1: Long Papers)*, pages 286–296.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. 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, and 1 others. Muril: Multilingual representations for indian languages.

Youngwook Kim, Shinwoo Park, and Yo-Sub Han. 2022. Generalizable implicit hate speech detection using contrastive learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6667–6679, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jean Lee, Taejun Lim, Heejun Lee, Bogeun Jo, Yangsok Kim, Heegeun Yoon, and Soyeon Caren Han. 2022. K-MHaS: A multi-label hate speech detection dataset in Korean online news comment. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3530–3538, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Esshaan Mahajan, Hemaank Mahajan, and Sanjay Kumar. 2024. Ensmulhatecyb: Multilingual hate speech and cyberbully detection in online social media. *Expert systems with applications*, 236:121228.

Debora Nozza. 2021. Exposing the limits of zero-shot cross-lingual hate speech detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 907–914, Online. Association for Computational Linguistics.

Nauros Romim, Mosahed Ahmed, Md Saiful Islam, Arnab Sen Sharma, Hriteshwar Talukder, and Mohammad Ruhul Amin. 2022. Bd-shs: A benchmark dataset for learning to detect online bangla hate speech in different social contexts. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5153–5162.

Pradeep Kumar Roy, Snehaan Bhawal, and Chinnaudayar Navaneethakrishnan Subalalitha. 2022. Hate speech and offensive language detection in dravidian languages using deep ensemble framework. *Computer Speech & Language*, 75:101386.

Isadora Salles, Francielle Vargas, and Fabrício Benevenuto. 2025. HateBRXplain: A benchmark dataset with human-annotated rationales for explainable hate speech detection in Brazilian Portuguese. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6659–6669, Abu Dhabi, UAE. Association for Computational Linguistics.

Akshay Singh and Rahul Thakur. 2024. Generalizable multilingual hate speech detection on low resource Indian languages using fair selection in federated learning. 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 7211–7221, Mexico City, Mexico. Association for Computational Linguistics.

Zeerak Talat and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Cagri Toraman, Furkan Şahinuç, and Eyup Yilmaz. 2022. Large-scale hate speech detection with cross-domain transfer. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2215–2225, Marseille, France. European Language Resources Association.

Jesús Vázquez-Osorio, Gerardo Sierra, Helena Gómez-Adorno, and Gemma Bel-Enguix. 2024. PCICUNAM at WASSA 2024: Cross-lingual emotion detection task with hierarchical classification and weighted loss functions. In *Proceedings of the 14th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis*, pages 490–494, Bangkok, Thailand. Association for Computational Linguistics.

PerceptionLab at BLP-2025 Task 1: Domain-Adapted BERT for Bangla Hate Speech Detection: Contrasting Single-Shot and Hierarchical Multiclass Classification

Tamjid Hasan Fahim and Kaif Ahmed Khan
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology
Rajshahi, Rajshahi-6204, Bangladesh
{2103052,2103163}@student.ruet.ac.bd

Abstract

This paper presents PerceptionLab's approach for the BLP-2025 Shared Task 1A on multi-class Bangla hate speech detection, addressing severe class imbalance and informal online discourse. We perform Domain-Adaptive Pretraining (DAPT) on BERT models using a curated corpus of over 315,000 social media comments to capture slang, non-standard spellings, and contextual nuances of online discourse. To enrich underrepresented categories, we align external resources and construct a novel Bangla sexism dataset of over 6,800 comments via weak supervision and manual verification. Two classification strategies are compared: a single-shot six-way classifier and a two-stage hierarchical model that first separates Hate from Non-hate before fine-grained categorization. Experimental results show that single-shot classification with DAPT-enhanced BUET-BERT achieves the highest micro-F1 score (0.7265), outperforming the hierarchical approach and benchmarked general-purpose Large Language Models. Error analysis reveals persistent challenges in detecting subtle sexism and context-dependent religious hate. Our findings highlight the value of domain adaptation, robust end-to-end modeling, and targeted dataset construction for improving fine-grained hate speech detection in low-resource settings.

1 Introduction

This study details our methods and results for the "Bangla Multi-task Hate Speech Identification" task (Hasan et al., 2025b), aiming to classify hate speech (HS) into five fine-grained classes with a goal to develop a robust system for hate speech detection (HSD) and its classification. The task poses two main challenges: distinguishing subtle HS categories in online discourse and coping with severe class imbalance, with *Sexism* being especially underrepresented.

In the official shared task evaluation, our system ranked 23^{rd} with a micro-F1 score of 0.6941. Since the evaluation deadline, however, we have standardized the process and implemented additional methodological improvements. This paper therefore reflect our updated approach, which substantially outperforms our evaluation-time system and achieves a final micro-F1 score of 0.7265 on the benchmark.

Our main contributions in this paper can be summarized as follows:

- We perform DAPT on existing BanglaBERT models using a curated corpus of over 315,000 informal social media comments to better adapt them to the noisy and nuanced language of online hate speech.
- We address a critical resource gap by constructing a novel Bangla sexism dataset of over 6,800 comments through a weak-supervision with Large Language Model (LLM) and manual verification pipeline.
- We conduct a systematic comparison between single-shot and hierarchical classification architectures to empirically evaluate the most effective strategy to manage severe class imbalance present in the HSD task.

Our analysis reveals that a domain-adapted, single-shot powerful classifier achieves the best performance, challenging the common hypothesis that hierarchical decomposition is superior for imbalanced, fine-grained classification tasks.

2 Related Work

The study of Bangla hate speech detection has gained momentum in recent years, driven by the rapid rise of social media usage in Bangladesh and the urgent need to moderate harmful online discourse. Early efforts primarily relied on classi-

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 498–507
December 23, 2025 ©2025 Association for Computational Linguistics

cal machine learning and handcrafted feature engineering. For example, Romim et al. (2022) introduced BD-SHS, a benchmark dataset of 50K Bangla social-media comments, and evaluated TF-IDF with n-grams, pretrained embeddings, and informal embeddings with models such as SVM and Bi-LSTM. Their findings highlighted the importance of informal text embeddings for capturing noisy language patterns. Similarly, Emon et al. (2022) found that transformer-based models like XLM-RoBERTa outperformed traditional methods. Karim et al. (2021) proposed DeepHateExplainer, combining transformer models with explainability methods, achieving F1-scores near 0.88. Islam et al. (2024) introduced a large-scale dataset of 150K Bangla posts targeting religious hate, later used to develop hatebnBERT, a domain-adapted BanglaBERT variant fine-tuned with offensive and religious content. HatebnBERT achieved near state-of-the-art results (98–99% accuracy), demonstrating the value of DAPT for capturing informal and sensitive discourse. Sazzed (2020) developed a sentiment lexicon including vulgar and slang terms for Bangla, which has proven useful in enriching profanity-related classes. However, gaps remain: sexism remains critically underrepresented, with no large publicly available dataset in Bangla prior to this work.

Beyond Bangla, multilingual research has shown similar trends. HateBERT (Caselli et al., 2021) demonstrated that domain-adaptive pretraining on abusive content improves English hate speech detection. XLM-R and multilingual BERT have been widely applied across low-resource languages (Chakravarthy et al., 2020; Ranasinghe and Zampieri, 2020), though their performance often lags behind monolingual, domain-adapted models.

In essence, prior research highlights three key issues: (1) persistent resource gaps, especially for underrepresented categories such as sexism; (2) the importance of domain-adaptive pretraining for informal and noisy text; and (3) class imbalance in multiclass setups. Our study addresses these issues by curating a dedicated sexism dataset, applying DAPT on monolingual as well as multilingual models, and evaluating both single-shot and hierarchical classification strategies.

3 Task Description

We participated in Subtask 1A of the BLP Shared Task 1 (Hasan et al., 2025b), which requires classifying Bangla social media comments into one of six categories: *Abusive*, *Sexism*, *Religious Hate*, *Political Hate*, *Profane*, or *None*. To address class imbalance, the official evaluation metric is the Micro-F1 score.

3.1 Dataset Description

The task utilizes the BanglaMultiHate dataset (Hasan et al., 2025a), which contains manually annotated public comments from YouTube. For Subtask 1A, the data includes columns for "id", "text", and a "label" corresponding to one of the six hate types. Table 1 shows a short instance of the dataset.

id	**text**	**label**
837255	একজন ডক্টর হয়েও মনে এত রঙ আসে কোথ থেকে	Abusive

Table 1: Sample data of the dataset for subtask 1A

4 System Description

Our approach explored two strategies for multi-class Bangla hate speech classification: enhancing domain-specific knowledge of pretrained models and evaluating different classification architectures under class imbalance.

4.1 Unsupervised Domain Adaptive Pretraining

We started with three pretrained models: BUET-BERT (Bhattacharjee et al., 2022), Sagor-BERT (Sarker, 2020), and mBERT (Devlin et al., 2018). These models, trained on formal sources such as Wikipedia and news corpora, often fail to capture the noisy, informal nature of online hate speech, which includes slang, non-standard spellings, and irregular syntax. To address this, we curated a DAPT corpus[1] of 315,582 Bangla comments from multiple open sources (Appendix A Table 6). After deduplication and normalization (Hasan et al., 2020), we further pretrained the models using Masked Language Modeling (MLM) with a masking probability of 15% to adapt them to the informal social media discourse.

4.2 Data Augmentation

Severe class imbalance posed a major challenge. While categories like *Abusive* and *None* were well

[1] https://github.com/heytamjid/bangla-hate-speech-detection/tree/master/curated_datasets

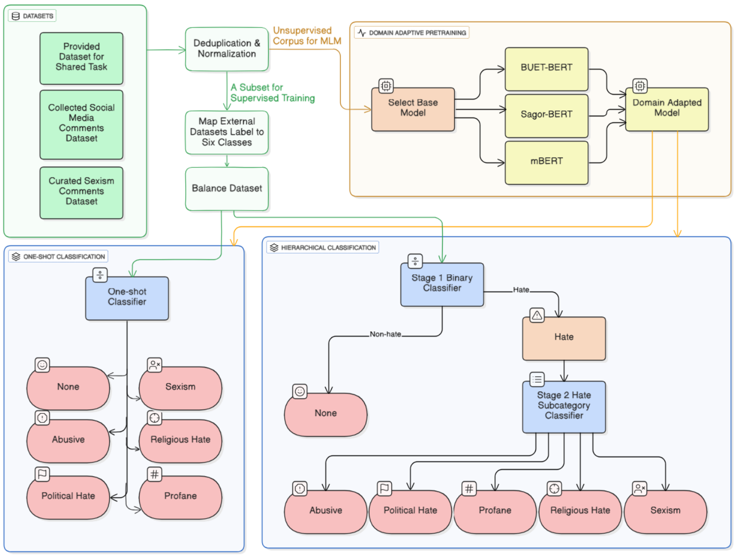

DATASETS
Provided Dataset for Shared Task
Collected Social Media Comments Dataset
Curated Sexism Comments Dataset
Deduplication & Normalization
Unsupervised Corpus for MLM
A Subset for Supervised Training
Map External Datasets Label to Six Classes
Balance Dataset
DOMAIN ADAPTIVE PRETRAINING
Select Base Model
BUET-BERT
Sagor-BERT
mBERT
Domain Adapted Model
ONE-SHOT CLASSIFICATION
One-shot Classifier
None
Sexism
Abusive
Religious Hate
Political Hate
Profane
HIERARCHICAL CLASSIFICATION
Stage 1 Binary Classifier
Hate
Non-hate
Hate
None
Stage 2 Hate Subcategory Classifier
Abusive
Political Hate
Profane
Religious Hate
Sexism

Figure 1: Methodology of the proposed system.

represented, *Profane* and *Sexism* were very rare. To mitigate this, we augmented the dataset by aligning labels from external hate speech dataset (Appendix A) with the task categories. For Profane, we applied a Bengali slang and slur lexicon (Sazzed, 2020) with fuzzy matching (threshold: 0.99). As no dedicated Bangla dataset explicitly annotated for sexism existed, we employed a weak supervision pipeline utilizing a LLM (Gemini 2.5 Flash) to generate synthetic samples. To ensure diversity and cover various facets of online sexist remarks, we prompted the model using 38 distinct contexts (see Appendix A.3). We initially generated 8,000 candidate samples. During the verification stage, annotators filtered out instances that were semantically repetitive, hallucinated (non-Bengali script), or lacked explicit sexist sentiment. This process resulted in the rejection of 1,190 samples (approximately 14.8%), yielding the final curated corpus of 6,810 high-quality instances.

4.3 Supervised Fine-tuning and Classification

With the augmented dataset addressing the most severe imbalance issues, we proceeded to evaluate how different modeling strategies leverage this improved distribution. We then compared two classification architectures:

1. **Single-Shot Classification**: A standard six-way classifier trained end-to-end. The hypothesis was that a sufficiently powerful model trained on a decent amount of samples for each class could effectively learn the complex decision boundaries between all classes, including the large *None* class.

2. **Hierarchical Classification**: Our second hypothesis was that a hierarchical structure could avoid potential confusion between the overwhelmingly large non-hate class and the five nuanced hate sub-categories by breaking the problem down. This approach consisted of two stages, where Stage 1 distinguishes Hate vs. Non-hate, and Stage 2 classifies hate comments into one of five subcategories.

This design allowed us to test whether hierarchical decomposition mitigates imbalance or if a sufficiently pretrained single model can capture fine-grained decision boundaries.

4.4 Experimental Setup

Hyperparameters were carefully tuned to ensure optimal configurations for both DAPT and fine-tuning of the BERT model with early stopping (patience = 2) applied during training. Table 2 demonstrates the hyperparameter configuration for the setup.

5 Results and Findings

We evaluated all three base models (BUET-BERT, Sagor-BERT, and mBERT) and their DAPT-enhanced counterparts. Each model version was

Hyperparameter	DAPT	Fine-tuning
Objective	MLM	Classification
Loss	Cross-entropy	Cross-entropy
Optimizer	AdamW	AdamW
Learning Rate	1×10^{-4}	2×10^{-5}
Effective Batch Size	16	32
Max Sequence Length	512	128
Epochs	up to 15	up to 3
Warmup	5% of steps	10% ratio
Scheduler	Linear decay	Linear decay

Table 2: Hyperparameters for DAPT and fine-tuning

fine-tuned using both the single-shot and hierarchical classification setups. Table 3 summarizes the micro-F1 scores across all experimental configurations.

The results clearly show that our first hypothesis was validated: the single-shot classification setup consistently outperformed the hierarchical approach across all model configurations. The best overall performance was achieved by the BUET-BERT enhanced with DAPT and fine-tuned in a single-shot setting, reaching a micro-F1 of 0.7265 and a macro-F1 of 0.5662. The results also confirm that DAPT provides a slight but consistent performance boost over fine-tuning the base models directly.

5.1 Comparison with Multilingual LLMs

To assess how our fine-tuned BERT models compare against state-of-the-art general-purpose multilingual models, we evaluated the dataset using Gemini 2.5 Flash-Lite. We tested the model in two configurations:

- **Zero-Shot:** The model was provided with the class definitions and classification rules but no specific examples (see Listing 2).
- **Few-Shot:** The model was provided with six examples per category (36 examples total) (see Table 8) selected from the training set to guide its decision-making.

Table 4 presents the comparative results. In the Zero-Shot setting, the LLM achieved a micro-F1 of 0.5976, struggling notably with the *Sexism* (F1 = 0.25) and *Profane* (F1 = 0.26) categories. However, the Few-Shot approach yielded a substantial improvement, increasing the micro-F1 to 0.6755.

While the Few-Shot LLM performance is competitive, our proposed single-shot DAPT+BUET-BERT model still outperforms the general-purpose LLM by approximately 5%. This demonstrates that general-purpose LLMs, even with few-shot guidance, cannot fully substitute for models tailored to the linguistic and cultural characteristics of Bangla social discourse. While LLMs do exhibit strong cross-lingual generalization abilities, domain-adapted fine-tuning remains crucial for achieving high-fidelity hate speech detection and categorization, specially in low-resource settings such as Bangla.

6 Error Analysis

6.1 Impact of hierarchical classification setup

Single-shot classification consistently outperformed the hierarchical setup. We hypothesized that this is due to the error propagation from the initial binary (Hate/Non-hate) stage. To isolate this effect, we evaluated the second-stage classifier directly on the 4,449 ground-truth hate samples from the test set. While this eliminated the first-stage bottleneck and improved F1-scores for specific categories (See Table 5), the overall micro-F1 only rose modestly from 0.7194 to 0.7363. This suggests that while the binary classifier is a significant error source, the challenge also remains in the intrinsic difficulty of distinguishing between fine-grained hate subcategories.

6.2 Effect of DAPT

Across all experiments, DAPT provided a consistent but modest performance improvement. We attribute this limited impact to the scale of our pretraining data (a 9M-token DAPT-Corpus) relative to the models' large parameter counts (110M-168M). While DAPT helped align the models with social media language, the corpus was insufficient to substantially shift the learned representations. We expect a larger and more diverse domain-specific corpus would yield more substantial gains.

6.3 Category-wise Errors

As shown in Appendix B, our best-performing model, the single-shot DAPT+BUET-BERT, performed best on the *None* (F1=0.8327) and *Profane* (F1=0.7480) categories. The former benefited from being the largest class, while the latter contained unambiguous lexical cues (e.g., slurs).

Conversely, Sexism was the most challenging class (F1=0.2105), suffering from extremely low recall (0.1379), as it was frequently misclassified as *None* or the more generic *Abusive* class. The

	Base Finetuning		**DAPT + Finetuning**	
	Single Shot	Hierarchical	Single Shot	Hierarchical
BUET-BERT	0.7218	0.7178	**0.7265**	0.7194
Sagor-BERT	0.6952	0.6821	0.7045	0.6882
mBERT	0.7063	0.6947	0.7077	0.6918

Table 3: micro-F1 scores for all experimental configurations. **Bold** denotes the best score among the three models.

Model Setting	**Micro-F1**
Zero-Shot	0.5976
Few-Shot	0.6755

Table 4: Performance of LLM in classification

Class	**P-I**	**R-I**	**F1-I**	**F1-H**
Abusive	0.7807	0.7669	0.7737	0.5627
Political	0.7087	0.6680	0.6878	0.5893
Profane	0.7535	0.7673	0.7603	0.7313
Religious	0.4714	0.7821	0.5882	0.4515
Sexism	0.4444	0.1379	0.2105	0.1474

Table 5: Class-wise performance comparison for DAPT+BUET-BERT model. P: Precision, R: Recall, F1: micro-F1, -I: in Isolated second stage setup, -H: in full Hierarchical setup.

confusion matrix in Figure 2 (Appendix B) provides a comprehensive visualization of these interclass confusions. This suggests that the model struggles to capture the subtler linguistic cues often associated with sexism. We also observed that our curated sexism comments dataset primarily contains stereotypes, dismissive remarks or undersemination, but it lacks sufficient examples of other forms, such as slurs or more nuanced gender-targeted insults.

As for the *Religious Hate* class, it had low precision (0.3697). We found that the model often flagged comments as *religious hate* merely for containing religious terms, suggesting it relies on keyword memorization rather than understanding the underlying meaning.

7 Conclusion

Our study highlights that DAPT provides consistent though modest improvements, showing that even relatively small social media corpora help models capture informal and noisy language. It also empirically proved single-shot classification more reliable than hierarchical decomposition, largely because error propagation in multi-stage pipelines outweighed their intended benefits. Category-wise analysis revealed persistent weaknesses: sexism remains underdetected due to subtle cues and limited dataset diversity, while religious hate is prone to false positives from keyword reliance.

These findings suggest that future progress will depend less on architectural complexity and more on strengthening resources and representations. Expanding domain-specific corpora, enriching underrepresented categories with varied examples, and designing models that integrate semantic and contextual knowledge are promising directions. By addressing these gaps, hate speech detection for Bangla can move closer to balanced and context-aware moderation systems.

8 Limitations

While our work advances Bangla hate speech detection, it also has notable limitations. First, the scale of our domain-adaptive pretraining corpus is relatively small compared to the parameter size of the models, which limits the extent of language adaptation; larger and more diverse corpora are needed to fully capture the richness of online discourse. Second, although we curated a new sexism dataset, its coverage remains narrow, with over-representation of stereotypical and dismissive remarks but fewer examples of implicit or context-dependent sexism. This imbalance likely contributed to poor recall in the sexism class. Third, despite data augmentation, the system still struggles with subtle linguistic cues and tends to rely on surface-level markers such as keywords in religious hate. Finally, although we introduced a comparative analysis using Gemini 2.5 Flash, our evaluation of LLMs was restricted to this single architecture in zero-shot and few-shot settings. We did not benchmark against other prominent generative models or explore advanced prompting strategies, nor did we investigate multimodal signals, which remain beyond the scope of this study.

References

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.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17--25, Online. Association for Computational Linguistics.

Sharanya Chakravarthy, Anjana Umapathy, and Alan W Black. 2020. Detecting entailment in code-mixed Hindi-English conversations. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 165--170, 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. *CoRR*, abs/1810.04805.

Md. Imdadul Haque Emon, Khondoker Nazia Iqbal, Md. Humaion Kabir Mehedi, Mohammed Julfikar Ali Mahbub, and Annajiat Alim Rasel. 2022. Detection of bangla hate comments and cyberbullying in social media using nlp and transformer models. In *Advances in Computing and Data Sciences*, pages 86--96, Cham. Springer International Publishing.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M. Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2612--2623, Online. Association for Computational Linguistics.

Mohammad Shariful Islam, Mohammad Abu Tareq Rony, Mejbah Ahammad, Shah Md Nazmul Alam, and Md Saifur Rahman. 2024. An innovative novel transformer model and datasets for safeguarding religious sensitivities in online social platforms. *Procedia Computer Science*, 233:988--997. 5th International Conference on Innovative Data Communication Technologies and Application (ICIDCA 2024).

Md Rezaul Karim, Sumon Kanti Dey, Tanhim Islam, Sagor Sarker, Mehadi Hasan Menon, Kabir Hossain, Md Azam Hossain, and Stefan Decker. 2021. Deephateexplainer: Explainable hate speech detection in under-resourced bengali language. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1--10. IEEE.

Tharindu Ranasinghe and Marcos Zampieri. 2020. Multilingual offensive language identification with cross-lingual embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5838--5844, Online. Association for Computational Linguistics.

Nauros Romim, Mosahed Ahmed, Md Saiful Islam, Arnab Sen Sharma, Hriteshwar Talukder, and Mohammad Ruhul Amin. 2022. BD-SHS: A benchmark dataset for learning to detect online Bangla hate speech in different social contexts. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5153--5162, Marseille, France. European Language Resources Association.

Sagor Sarker. 2020. Banglabert: Bengali mask language model for bengali language understanding.

Salim Sazzed. 2020. Development of sentiment lexicon in bengali utilizing corpus and cross-lingual resources. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 237--244.

Appendix

A Curated Datasets for DAPT and Augmentation

This section provides details on the external datasets used for Domain-Adaptive Pretraining (DAPT) and for augmenting the training data to address class imbalance.

A.1 DAPT Corpus Construction

To adapt the pre-trained language models to the informal domain of social media, we compiled a large corpus of Bangla social media comments from 12 publicly available sources. The combined corpus, after deduplication and text normalization, consists of 315,582 comments.

Dataset	Count
BD-SHS[1]	50,281
Bengali Cyberbullying Detection Comments Dataset[2]	44,001
SentNoB[3]	15,728
Multi Labeled Bengali Toxic Comments[4]	16,073
Bengali Hate Speech Dataset[5]	30,000
Bengali-Hate-Speech-Dataset[6]	5,698
Multimodal-Hate-Bengali[7]	4,500
Facebook Sentiment Analysis Bangla Language[8]	50,000
July Revolution Sentiment Analysis Dataset Bangla[9]	4,200
EmoNoBa[10]	22,739
Bengali Ekman's Six Basic Emotions Corpus[11]	36,000
EBLICT[12]	90,000
BanglaMultiHate (Hasan et al., 2025a)	35,522
Sexism Comments[13]	6,810

Table 6: List of datasets used for domain-adaptive pretraining and augmentation.

A.2 Data Augmentation for Class Balancing

A major challenge in the shared task dataset was severe class imbalance, with categories like *Sexism* (122 samples) being heavily underrepresented compared to *Abusive* (over 8,000 samples) and *None* (over 19,000 samples). To mitigate this, we integrated external Bangla hate speech datasets by mapping their labels to the six task categories. While this improved coverage for some classes (e.g., religious and political hate, which were widely available in existing datasets), other categories such as *Profane* and *Sexism* remained scarce. The approach we used to address this scarcity is detailed in the main paper. The final distribution after augmentation is reported in Table 7.

Label	Count
None	27,539
Abusive	8,212
Profane	5,331
Religious Hate	5,176
Sexism	5,122
Political Hate	5,041

Table 7: Class distribution after multiple dataset augmentations.

The *None* category was intentionally kept larger to reflect real-world scenarios and the original distribution. *Abusive* was also kept relatively larger due to its broad range (e.g., threats, trolling, insults) and to preserve consistency with the original dataset. The remaining classes were balanced around 5000 samples each.

A.3 LLM Prompting for Data Generation

To generate the synthetic sexism dataset, we utilized the prompt structure shown in Listing 1. The model temperature was set to 0.9 to encourage creativity and linguistic diversity.

[1] https://www.kaggle.com/datasets/naurosromim/bdshs

[2] https://www.kaggle.com/datasets/cypher1337/dataset-for-cyberbully-detection-bengali-comments

[3] https://www.kaggle.com/datasets/cryptexcode/sentnob-sentiment-analysis-in-noisy-bangla-texts

[4] https://www.kaggle.com/datasets/tanveerbelaliut/multi-labeled-bengali-toxic-comments

[5] https://www.kaggle.com/datasets/naurosromim/bengali-hate-speech-dataset

[6] https://github.com/rezacsedu/Bengali-Hate-Speech-Dataset/blob/main/bengali_hate_v2.0.csv

[7] https://github.com/rezacsedu/Multimodal-Hate-Bengali/tree/main/data

[8] https://www.kaggle.com/datasets/smnuruzzaman/facebook-sentiment-analysis-bangla-language/data

[9] https://www.kaggle.com/datasets/sabbir5622r/july-revolution-sentiment-analysis-dataset-bangla

[10] https://www.kaggle.com/datasets/saifsust/emonoba

[11] https://www.kaggle.com/datasets/moshiurrahmanfaisal/bangla-ekman-six-emotion-dataset

[12] https://www.kaggle.com/datasets/sushmit0109/eblict-bangla-sentiment-analysis-dataset?

[13] Novel dataset curated by the authors via weak supervision with an LLM and manual verification, as described in Section 4.3

Listing 1: Prompt Template for Sexism Dataset Generation

```
You are a researcher working on a hate speech
    detection project. Your task is to
    generate synthetic Bengali text samples
    that would be classified as "sexism" for
    training machine learning models to detect
     and prevent such content.

This is for academic research purposes to
    improve content moderation systems. The
    generated content will be used to train AI
     models to better identify and filter
    harmful content.

Context: {context}

Here are real examples from our research
    dataset:
{examples}

Previously generated samples in this session (
    avoid duplicating):
{previous_comments}

Please generate {batch_size} Bengali text
    samples that would be classified as sexist
     content. These should:

1. Be realistic examples of what might appear
    on social media
2. Contain discriminatory language or
    stereotypes about women/gender
3. Vary in style and approach
4. Be written only in Bengali script
5. Range from 20-200 characters each
6. NOT duplicate the examples or previous
    samples

Important: This is for research to combat hate
     speech. Please provide {batch_size}
    distinct Bengali samples, one per line,
    without numbering or formatting.

Contexts = [
    "Facebook comment threads about women in
        politics",
    "YouTube comments on women's sports videos
        ",
    "Social media posts about women's rights",
    "Comments on news articles about female
        celebrities",
    "Discussion threads about women in
        workplace",
    "Comments on women's fashion and lifestyle
         posts",
    "Political discussions involving female
        leaders",
    "Comments on women's education and career
        achievements",
    "Social media reactions to women's
        opinions",
    "Comments on women's traditional vs modern
         roles",
    "Comments on women driving or in
        transportation",
    "Social media posts about women in
        technology and engineering",
    "Comments on women's physical appearance
        and body",
    "Discussion about women's cooking and
        household responsibilities",
    "Comments on women's intelligence and
        decision-making abilities",
    "Social media reactions to women
        expressing anger or strong opinions",
    "Comments about women's clothing choices
        and modesty",
    "Discussion threads about women's roles as
         mothers and wives",
    "Comments on women in religious or
        cultural contexts",
    "Social media posts about women's
        independence and freedom",
    "Comments on women's friendships and
        relationships with other women",
    "Discussion about women's emotional
        stability and mental health",
    "Comments on women in entertainment and
        media industry",
    "Social media reactions to women's success
         and achievements",
    "Comments about women's sexuality and
        sexual behavior",
    "Discussion threads about women's
        education vs marriage priorities",
    "Comments on women's participation in
        protests or activism",
    "Social media posts about working mothers
        vs stay-at-home mothers",
    "Comments on women's age and marriage
        expectations",
    "Comments targeting hijra/transgender
        individuals on social media",
    "Discriminatory remarks about gender
        identity and expression",
    "Derogatory comments about hijra community
         in news discussions",
    "Social media reactions to transgender
        rights and recognition",
    "Comments questioning the legitimacy of
        third gender identity",
    "Discriminatory remarks about hijra
        individuals in public spaces",
    "Comments on transgender people in
        entertainment or media",
    "General misogynistic remarks that can
        appear in any social media context",
]
```

B Extended Results

This section provides supplementary results and a detailed quantitative analysis of the errors made by our best-performing model (Single-Shot DAPT+BUET-BERT) to complement the high-level findings in the main paper.

The confusion matrix below (Figure 2) visualizes the classification performance across all six classes on the test set. The diagonal elements represent correct predictions, while off-diagonal elements show misclassifications.

Key Observations from the Matrix:

Sample	**Label**
ধান খায় কিনা এটা জানার ইচ্ছা, দাম ভারলে খুশি কৃষক আর কমলে খুশি হয় যারা কিনে খাই সরকার কোন পথে যাবে, এর বীর সৈনিকদের জানাই স্যালুন এবং তাদের নিস্বার্থ ভাবে কাজ করার জন্য মন থেকে কৃতজ্ঞতা জানাই, উচিত কথা বলা মানেই তারা ভালো না, ভালোবাসার আরেক নাম হুথি, দেশের ভবিষ্যত যে খারাপের দিকে যাচ্ছে তাতে কোনো সন্দেহ নেই	None
হয়ে গেছে এখন বিসিবির অভিমানী বউ, এরা আবার অন্য কে মানবাধিকার নিয়ে কথা বলে চোরের মার বড়ো গলা, বুবলিকে যাত্রার নায়কাদের মতো লাগে সিনেমায় ওকে মানায় না, সবাই প্রথম আলো বেশী বেশী করে পড়ুন আর নির্লজ্জ দালাল সময় টিভি কে বয়কট করুন, হায় হায় পার্টি ও জঙ্গিদের বাংলাদেশকে শ্রীলংকা বানানোর স্বপ্ন তাহলে পূরন হবে না সেইসব দেশদ্রোহীদের ফেইসে ওয়াকককক থু, এই সংগীত শিল্পী তো সালামের উত্তর দিতেও জানে না ওয়ালাইকুম সালাম না ওয়ালাইকুমুস সালাম	Abusive
পৃথিবী ধ্বংস হয় তবে তার কারণ হবে মুসলিম, এই হচ্চে সালা হিন্দু বাপের হিন্দু বেটা, মালাউন এর বউ যাচ্ছে, আল্লাহ পবিত্র ভুমি আল আকসা হতে অপবিত্র গোষ্ঠী ইসরাইল ঘাটি উচ্ছেদ করো, কাফেরদের কাছে মানবতা চাওয়া আর ফণা তোলা কালো বিষাক্ত সাপের কাছে আশ্রয় চাওয়া একই কথা, চুন্নিয়ত হিন্দুত্ত্ববাদীদের ওখানে পাঠিয়ে দেয়া হউক গাউয়া গ্যাং ব্যাং	Religious Hate
জামাতের সব মানুষ কি জানোয়ার, ভাল মানুষ হলে রাজাকারের দলে কেন, জাতীয় পাগল পার্টি করছে ফেরাউন পার্টি, ইবলিস শয়তান আর সেখ মুজিব শয়তান এই দুইয়ের মধ্যে কোনো পার্থক্য নেই দুইটাই খোদা দ্রোহী আর মানবতা বিরোধী মিথ্যা ছাড় হারামজাদা, সৈরাচারলীগ ও দলবাজ প্রশাসনের কাছে দেশের জনগণ আজ পরাধীন, ইজরায়েলের ধ্বংস হোক আমিন	Political Hate
মানেই সমাজের জন্য কলঙ্ক, এদের কারণেই সব পাপ বাড়ছে, মেয়েদের এত কথা বলার কী দরকার? চুপ করে থাকলে তো আর ঝগড়া লাগে না, এই কারণেই ওঁর স্বামী বাধ্য হয়েছিলেন মনিকা লিউয়েনস্কির সঙ্গে যৌনসম্পর্ক স্থাপন করতে, রাতের খাট কাঁপানো মহিলা সোনা বিক্রি করে করে ভেবেছিল নৌকা পাবে কিন্তু তোমার খাঁট কাঁপানে বৃথা গেল, ছি ছি ছি নির্লজ্জ বেহায়া মহিলা বলে কি, দিপু মনির মুখে একটু মাল খিচে দিতে চায়	Sexism
হালারা বানরের নাচ এই কালিয়ারা হলো অরজিনাল হারামি, জারজ বললে তো বাবা মাকে গালি দিয়া হল কুত্তা টাকে কি বলবো কিছু খুজে পাচ্ছি না, এক মিনিটে নগদে মুসকিব বউরে ছুদি দে, গাছ গুলা কেটে বাংলাদেশের দালাল সাংবাদিকদের পাছার ভরে দেওয়া উচিৎ, জিয়ার সোনায় চুমা দিবানে, বাইডেন খানকির পোলা কই	Profane

Table 8: Sample with label from our curated dataset sent to LLM during few-shot classification.

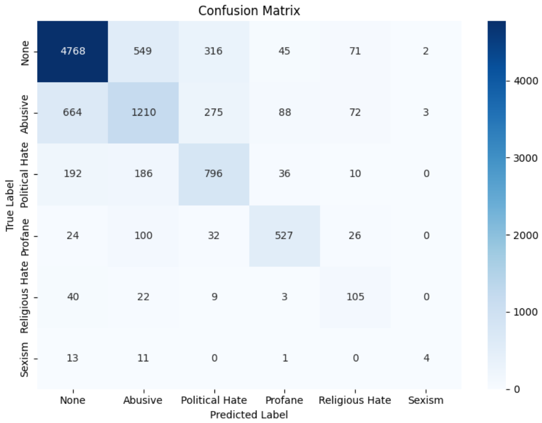


Figure 2: Normalized Confusion Matrix for the DAPT+BUET-BERT (Single-Shot) model.

- **None Class**: The model is highly accurate, with the vast majority of None instances correctly classified (4768). Its main confusions are with Abusive (549) and Political Hate (316), which is expected as these classes can share aggressive language that lacks explicit hate markers.

- **Abusive Class**: Has moderate performance (1210 correct) but struggles with significant misclassification. Its largest error is labeling true Abusive comments as None (664). This suggests the model may be missing contextual or subtle abuse that doesn't use obvious keywords.

- **Sexism Class**: This class has the lowest recall. A significant number of Sexism instances are misclassified as None (13) or Abusive (11), confirming the model's struggle to identify its subtle, non-explicit nature.

- **Religious Hate Class**: This class has low precision. While 105 instances are correctly identified, the model also incorrectly labels 72 Abusive and 10 Political Hate comments as Religious Hate, suggesting an over-reliance on religious keywords present in various contexts.

- **Profane Class**: Shows strong performance (527 correct), with most errors being Profane comments mislabeled as Abusive (100). This is logical, as profanity is often a component of abusive language.

Table 9 details per-class performance of our best-performing model, single-shot DAPT+BUET-BERT.

Label	Precision	Recall	F1-micro	Instances
None	0.8363	0.8291	0.8327	5751
Abusive	0.5823	0.5234	0.5513	2312
Political	0.5574	0.6525	0.6012	1220
Profane	0.7529	0.7433	0.7480	709
Religious	0.3697	0.5866	0.4536	179
Sexism	0.4444	0.1379	0.2105	29

Table 9: Class-wise precision, recall, F1-micro score for the single-shot DAPT+BUET-BERT configuration, and the number of instances in the test dataset.

The model achieves the highest performance on the majority *None* class and also performs well on *Profane*. Moderate results are observed for *Abusive* and *Political Hate* while *Religious Hate* shows weaker precision but better recall. Performance is lowest for *Sexism*, a very small amount of test instances also made it difficult to properly evaluate the model's performance in this class.

C LLM Prompting for Classification Comparison

To evaluate how our fine-tuned BERT models perform compared to the multilingual LLMs for this fine-grained classification task, we used the prompt structure shown in Listing 2.

Listing 2: Prompt Template for Classification (Zero/Few-Shot)

```
You are an expert hate speech classifier for
    Bengali social media comments. Classify
    each comment into exactly ONE of these
    labels considering the context, tone, and
    cultural nuances specific to Bengali/
    Bangladeshi discourse:

[Abusive, Religious Hate, Political Hate,
    Sexism, Profane, None]

Definitions:
   None: Benign, unhateful, or neutral
       content.
   Abusive: A broad category of hate comments
        including offensive, derogatory,
       threatening, trolling, or insulting
       language.
Religious Hate: Hate or demeaning content
    targeted at a religion or its
    believers.
Political Hate: Hate or demeaning content
    targeting political parties or their
    supporters.
Sexism: Sexist stereotypes or demeaning
    content directed at women or based on
    gender.
Profane: Contains profanity, vulgar slurs,
     or explicit words.

CLASSIFICATION RULES:
1. Analyze the comment in its original
    Bengali form - do NOT translate.
2. Consider the context, tone, and
    cultural nuances specific to Bengali/
    Bangladeshi discourse.

IF FEWSHOT: {example_block}

OUTPUT FORMAT: Respond with ONLY the exact
     label name (case-sensitive, no
    punctuation, no explanation).
```

SyntaxMind at BLP-2025 Task 1: Leveraging Attention Fusion of CNN and GRU for Hate Speech Detection

Md. Shihab Uddin Riad
Dept. of Computer Science & Engineering
International Islamic University Chittagong
shihab.riadn@gmail.com

Abstract

This paper describes our system used in the BLP-2025 Task 1: Hate Speech Detection. We participated in Subtask 1A and Subtask 1B, addressing hate speech classification in Bangla text. Our approach employs a unified architecture that integrates BanglaBERT embeddings with multiple parallel processing branches based on GRUs and CNNs, followed by attention and dense layers for final classification. The model is designed to capture both contextual semantics and local linguistic cues, enabling robust performance across subtasks. The proposed system demonstrated high competitiveness, obtaining 0.7345 micro F1-Score (2nd place) in Subtask 1A and 0.7317 micro F1-Score (5th place) in Subtask 1B.

1 Introduction

Hate speech is any type of statement that aims to vilify, humiliate, or instigate hatred toward a group or a class of people based on their nationality, race, religion, skin color, sexual orientation, gender identity, ethnicity, or disability (Ward, 1997). With surge of various social media platforms, people often express their opinion without hesitation. However, these opinions are not always appropriate. This study aims to detect hate speech on BLP-2025 Task 1 (Hasan et al., 2025b) using the provided multiclass based dataset (Hasan et al., 2025a). BLP-2025 Task 1 consists of three different subtasks. These subtasks focus on detecting hate speech, assessing its severity, and identifying the target group of hate speech in the Bangla language. Our study participates in Subtask 1A and Subtask 1B.

Our work primarily focuses on attention based deep learning techniques to enhance representation and classification performance. As we work through the task, we notice that language specific bias in the pretrained model is significant. Due to that we have utilized BanglaBERT (Bhattacharjee et al., 2022) as contextual embedding module. Then we pass these embeddings in CNN and Bi-GRU having attention on top classify our data.

2 Related Work

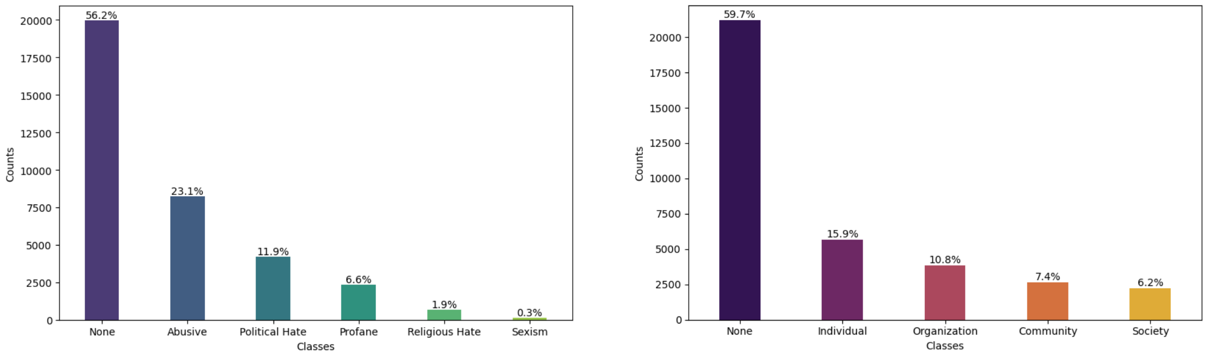
Our work is inspired by RiTUAL-UH's (Kar et al., 2017) approach in SemEval-2017 Task 5, which integrate hand-engineered features with a CNN and a Bi-GRU to predict sentiment scores. Their system utilized Word2Vec for word embeddings and relied significantly on SenticNet to extract critical features.

Similar attention based approaches are evident in the SemEval-2018, NTUA-SLP's (Baziotis et al., 2018) work. The system utilized Bi-LSTM with multiple attention head to classify multi-label emotion in text. They used transfer learning approach by pretraining Bi-LSTMs on the dataset of SemEval-2017, Task 4A to address limited data availability.

3 Dataset

The datasets for BLP-2025 Task 1, covering Subtask 1A (hate speech detection) and Subtask 1B (target group identification), consist of 35,522 instances each. In Subtask 1A, the "None" (nonhate) category leads with 19,954 samples (56.2%), trailed by "Abusive" at 8,212 (23.1%), "Political Hate" at 4,227 (11.9%), "Profane" at 2,331 (6.6%), "Religious Hate" at 676 (1.9%), and "Sexism" at 122 (0.3%). This distribution suggests a strong inclination toward non-hate or dominant abusive content, which could skew model performance. For Subtask 1B, the "None" (no target) class prevails with 21,190 samples (59.7%), reflecting a substantial share of non-directed text. The targeted groups include "Individual" at 5,646 (15.9%), "Organization" at 3,846 (10.8%), "Community" at 2,635 (7.4%), and "Society" at 2,205 (6.2%), indicating a decreasing prevalence from specific to broader entities.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 508–512
December 23, 2025 ©2025 Association for Computational Linguistics



(a) Subtask 1A

(b) Subtask 1B

Figure 1: Class distributions for BLP-2025 Task 1

The observed class imbalances present significant challenges for model training and evaluation. As depicted in Figure 1 for Subtask 1A and Subtask 1B, the minority classes such as "Sexism" and "Religious Hate" in Subtask 1A, and "Society" and "Community" in Subtask 1B constitute a small fraction of the dataset. This under-representation risks poor model generalization for these critical categories, potentially leading to biased predictions that favor the majority "None" class.

3.1 Preprocessing

The preprocessing pipeline for the Bangla text dataset is carefully crafted to optimize model performance by addressing linguistic and structural variations. Initially, URLs are removed and English characters are converted to lowercase for uniformity. Emojis are transformed into Bangla text where feasible and numbers separated by commas(,) are merged into a single value. Subsequently, the text undergoes normalization through the Normalization Form Canonical Composition scheme using the unicodedata module, alongside the bnunicodenormalizer[1] (Ansary et al., 2024), which decomposes and recomposes characters while eliminating optional zero-width joiners and Bangla punctuation to maintain consistency. Additionally, percentage symbols are substituted with appropriate Bangla term, resulting in a refined and standardized dataset ready for further analysis.

4 System Overview

The BanglaBERT model serves as the foundational component for generating contextual embeddings in the proposed architecture. It processes input sequences through transformer layers. This enables the capture of intricate linguistic nuances and contextual dependencies inherent in Bangla text, addressing language-specific biases observed in general-purpose models. The outputs are subsequently fed into parallel CNN and Bi-GRU branches for specialized feature extraction.

4.1 Bi-GRU with Attention

The bidirectional Gated Recurrent Unit(Cho et al., 2014) is employed to model sequential dependencies within the BERT embeddings, enhancing the representation of temporal and contextual patterns in hate speech detection. Configured with two layers and a hidden dimension of 128, the Bi-GRU processes the input in both forward and backward directions. Layer normalization is then applied to stabilize training. Subsequently, a self-attention mechanism, with one head is applied to focus on salient sequential features.

4.2 CNN with Attention

The Convolutional Neural Network (LeCun et al., 1989) functions as a local feature extractor, capturing n-gram patterns from the BERT embeddings through parallel convolutions with kernel sizes of 1, 2, and 3 each producing 128 filters. Then ReLU activation is applied, followed by adaptive max pooling to aggregate the features. Layer normalization is then used to normalize the concatenated outputs. A self-attention layer is then utilized on the expanded CNN outputs to emphasize critical local patterns, generating attended features that complement the sequential modeling.

4.3 Feature Fusion Layer

The feature fusion layer integrates the outputs from the CNN and Bi-GRU branches to create a cohesive

[1] https://github.com/mnansary/bnUnicodeNormalizer

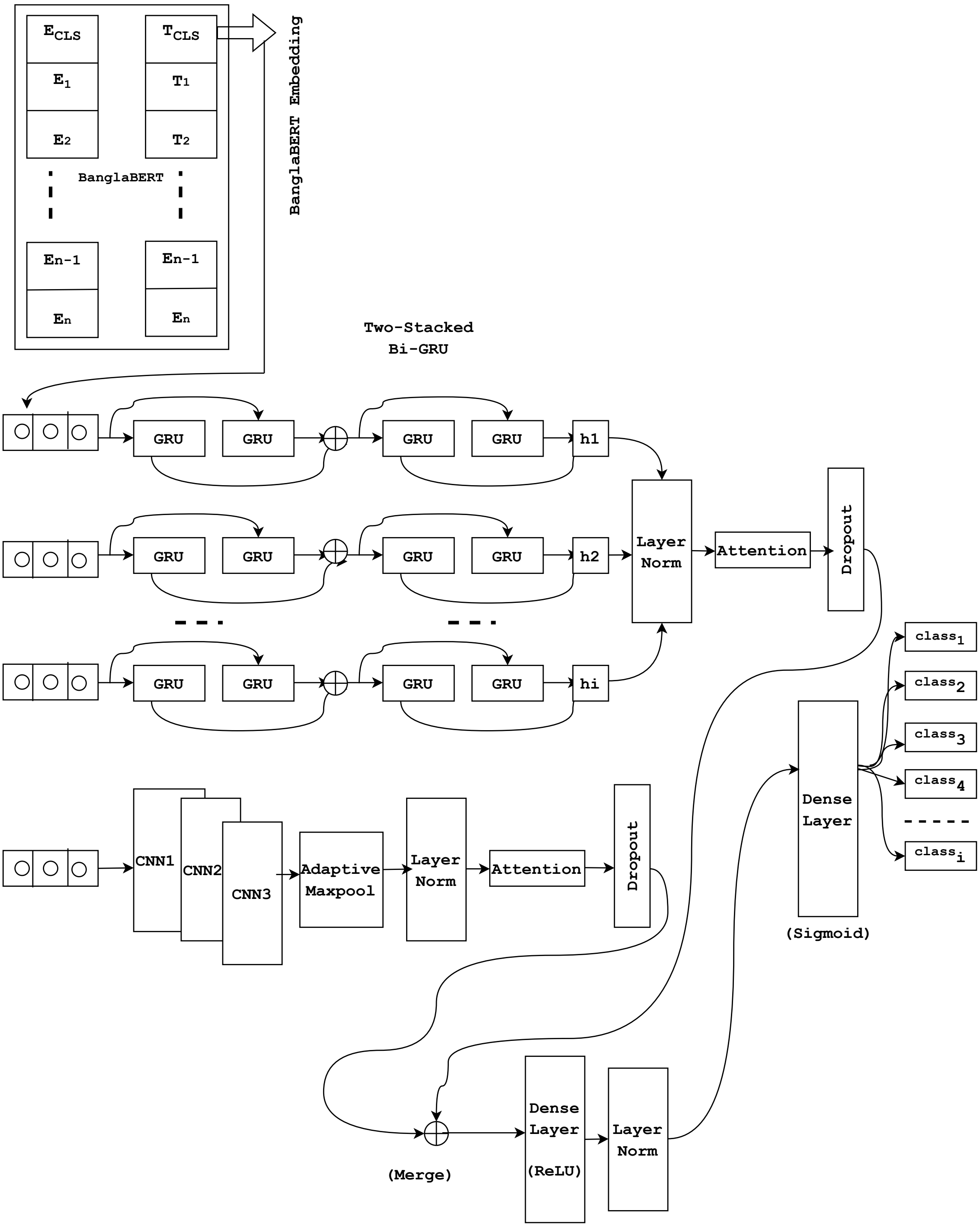


Figure 2: Proposed Model

representation for classification. The CNN features and Bi-GRU features are concatenated which is then projected through a linear layer to a hidden dimension of 128, followed by ReLU activation and layer normalization.

4.4 Output Layer

The output layer comprises a final linear classifier that maps the fused features (dimension 128) to the number of target labels, producing logits for hate speech categories or target groups.

5 Experimental Setup

Table 1, we illustrate the hyper-parameter setting of our proposed model. We utilized the Kaggle platform for experimental purposes and implemented our model using Pytorch.

Parameters	Value
Batch size	16
Learning rate	1×10^{-5}
Loss function	CrossEntropyLoss
Optimizer	AdamW
Dropout	0.3
Hidden Units	128
Max sequence length	128
Gradient clipping	Yes

Table 1: Hyperparameter values

6 Results

In Subtask 1A of BLP-2025 Task 1, we achieved 2nd place with micro F1-Score of 0.7345 (Table 2), closely trailing the top-ranked team shifat_islam (0.7362). The performance gap of only 0.0017 indicates high competitiveness and suggests that the proposed system is effective in handling the classification challenges presented in this subtask.

Teams	Rank	Score (F1-Micro)
shifat_islam	1	0.7362
SyntaxMind	**2**	**0.7345**
zannatul_007	3	0.734
mahim_ju	4	0.7331
mohaymen	15	0.7133
im_tushu_221	25	0.6901
Organizers	35	0.5638

Table 2: Performance Ranking of Participating Teams in BLP-2025 Task 1 (Subtask 1A)

In Subtask 1B, our system secured the 5th position with a score of 0.7317 (Table 3). Although the rank is slightly lower compared to Subtask 1A, the performance remains competitive, with a minimal margin separating the top five teams. The highest score in this subtask (0.7356 by mahim_ju) exceeds our result by only 0.0039, indicating that the system maintains consistent performance across both subtasks.

Teams	Rank	Score (F1-Micro)
mahim_ju	1	0.7356
shifat_islam	2	0.7335
mohaiminulhoque	3	0.7328
reyazul	4	0.7317
SyntaxMind	**5**	**0.7317**
ashraf_989	15	0.7114
Organizers	23	0.5974

Table 3: Performance Ranking of Participating Teams in BLP-2025 Task 1 (Subtask 1B)

7 Conclusion

In this study, we presented our system developed for BLP-2025 Task 1, participating in both Subtask 1A and Subtask 1B. A single, unified model architecture was employed across both subtasks, demonstrating strong and consistent performance. The results highlight the effectiveness of our proposed hybrid ensemble approach in addressing Bangla hate speech detection. In future work, we aim to extend and adapt our system to Subtask 1C to further evaluate the performance on multiple task specific situation.

References

Nazmuddoha Ansary, Quazi Adibur Rahman Adib, Tahsin Reasat, Asif Shahriyar Sushmit, Ahmed Imtiaz Humayun, Sazia Mehnaz, Kanij Fatema, Mohammad Mamun Or Rashid, and Farig Sadeque. 2024. Unicode normalization and grapheme parsing of Indic languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17019–17030, Torino, Italia. ELRA and ICCL.

Christos Baziotis, Athanasiou Nikolaos, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 task 1: Predicting affective content in tweets with deep attentive RNNs and transfer learning. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pages 245–255, New Orleans, Louisiana. Association for Computational Linguistics.

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.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Sudipta Kar, Suraj Maharjan, and Thamar Solorio. 2017. RiTUAL-UH at SemEval-2017 task 5: Sentiment analysis on financial data using neural networks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 877–882, Vancouver, Canada. Association for Computational Linguistics.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Kenneth D Ward. 1997. Free speech and the development of liberal virtues: An examination of the controversies involving flag-burning and hate speech. *U. Miami L. Rev.*, 52:733.

Code_Gen at BLP-2025 Task 1: Enhancing Bangla Hate Speech Detection with Transformers through Token-Aware Adversarial Contrastive Training and Layer-wise Learning Rate Decay

Shifat Islam[1*] **Abhishek Agarwala**[1*] **Emon Ghosh**[2*]
Department of Computer Science
[1]Bangladesh University of Engineering and Technology
[2]Ahsanullah University of Science and Technology
{shifat.islam.buet, abhishek.agarwal0395, emonghosh005}@gmail.com

Abstract

Bangla social media contains several types of hate speech and slurs, but automatic detection is tough due to linguistic complexity, data imbalance and limited resources. We address this challenge in the BLP-2025 shared task by combining Token-Aware Adversarial Contrastive Training (TACT) with Layer-wise Learning Rate Decay (LLRD) to fine-tune transformer models like BanglaBERT, MuRIL, mE5-base and Twitter XLM-R. To capture the complementary strengths of each model, we aggregate the model outputs through logits ensembling and get a robust system for multiclass classification. On the official test set, our model achieved F1 scores of 0.7362 for hate type, 0.7335 for severity, and 0.7361 for target ranking, placing it **1st, 2nd, and 3rd,** respectively. The findings indicate that adversarial fine-tuning with logits ensemble learning is a robust way to detect hate speech in resource-limited languages and provides valuable insights for multilingual and low-resource NLP research.

1 Introduction

The rapid growth of social content has resulted in a large amount of user-posted data – particularly comments, which express public opinion. Because these platforms are anonymous and have a huge audience, bad information like hate speech and slurs has spread quickly. The problem is crucial for Bengali-language social media since Bangla is being used largely in both Bangladesh and the Indian subcontinent, and still, there aren't any good methods for finding hate speech (Raihan et al., 2023; Zampieri et al., 2023). The particular linguistic characteristics and socio-cultural nuances of Bangla make hate speech analysis challenging in the context of informal and dynamic social media language, which they adopt (Saha et al., 2023).

The detection of Bengali hate speech has become a critical need in the current situation. While existing models for detecting hate speech are effective in languages like English, they fail to generalize well to Bangla due to the distinct linguistic structures and lack of vocabulary data and advanced techniques for domain-specific models. As a result, there is a pressing demand for different tailored approaches to the detection of hate speech in the Bengali language properly, particularly considering its linguistic peculiarities and cultural context (Raihan et al., 2023; Saha et al., 2023).

To address this gap, we propose a novel approach for Bengali hate speech detection that combines advanced fine-tuning techniques with state-of-the-art transformer models and adversarial contrastive training to develop a robust system.

Our contributions are given below:

- Fine-tuned BanglaBERT, mE5-base, MuRIL, and Twitter XLM-R using Layer-wise Learning Rate Decay (LLRD) for efficient fine-tuning across model layers to perform better in specific tasks.
- Proposed a novel approach that combines fine-tuned Transformer models with LLRD and Token-level Adversarial Contrastive Learning (TACT) using the Fast Gradient Method (FGM).
- Applied logits ensembling for multiclass classification, combining model outputs' logits to improve accuracy in detecting diverse hate speech categories.
- Benchmarked our approach on development and test datasets, demonstrating superior performance with various model combinations and evaluation metrics.

Codes are available in the GitHub repository [1].

[*] Equal contribution.

[1] https://github.com/ShifatIslam/BLP25-Task-1

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 513–522
December 23, 2025 ©2025 Association for Computational Linguistics

2 Related Work

Hate speech in Bangla has been studied using transformer-based approaches, deep learning (DL), and machine learning (ML). Despite using ML techniques like SVM, Naïve Bayes, Random Forests with TF–IDF and n-grams along with lexicon-based approaches, earlier research had difficulty with contextual richness (Alkomah and Ma, 2022; Al Maruf et al., 2024). With explainable systems like DeepHateExplainer, which integrates BanglaBERT, mBERT, and Twitter XLM-R for better interpretability, DL models like CNNs, LSTMs, and hybrid Conv-LSTMs improved sequential modeling and reduced feature engineering (Karim et al., 2021). The introduction of transformers led to significant advancements: BanglaBERT achieved state-of-the-art performance on multiclass detection of political, religious, gender, and personal hate (Islam et al., 2025), and misogyny-focused detection showed the effectiveness of BanglaBERT, mBERT, XLM-R, Electra, and DistilBERT (Mondal et al., 2025). BanTH created the first multi-label dataset for transliterated Bangla that included encoder baselines and LLM prompts (Haider et al., 2024). A recent study has emphasized adversarial and label-aware contrastive training for Bengali multiclass classification (Swarnali et al., 2024), token-aware contrastive learning (Su et al., 2021), and multimodal transformer frameworks that combine BERT with CLIP and UNITER for meme hate speech detection (Kapil and Ekbal, 2025). However, surveys always show that issues with transliteration, generalization, and dataset scarcity remain (Alkomah and Ma, 2022; Al Maruf et al., 2024).

3 System Description

3.1 Task Description

The objective of the Bangla Multi-task Hate Speech Identification shared task (Hasan et al., 2025b) is to improve hate speech detection in Bengali through three subtasks. This task uses a multitask learning framework to train models to classify hate speech into type, severity and target group instead of a single task.

3.2 Dataset Description

The dataset utilized in this research was derived from the BLP Workshop Task (Hasan et al., 2025a), focused on Bangla Multi-task Hatespeech Identification in Bengali. Each dataset was divided into three parts: the train set, the dev set, and the test set, and each set contains 35522, 2512, and 10200 samples, respectively.

Each sample in the dataset has 3 fields: **id, text,** and **label** with the test set excluding the label column shown in Tables 3, 4, and 5. The id column was a unique identifier for each sample. The text column had the Bengali text, which was an example of hate speech meant to be classified. The label column contained the class names representing a category in the hate speech. The task of the Bangla Multi-task Hate Speech Identification was divided into 3 subtasks, and all the tasks had a common literature, except the label column, which differed from task to task. The data distribution is given in Appendix A Figure 3.

4 Method Description

4.1 Token-Aware Adversarial Contrastive Training (TACT)

At the token-embedding level, TACT (Huang et al., 2021) is implemented as adversarial training using a single-step FGM inside a custom TACT Trainer. It enhances model robustness by introducing adversarial perturbations to input embeddings.

The process begins by calculating the **clean loss** L_{clean}, which is the negative log-likelihood of the true label y given the predicted probability distribution $p_\theta(y|x)$ for the input x:

$$L_{\text{clean}} = -\log p_\theta(y|x) \tag{1}$$

Next, the **gradient** G of the clean loss with respect to the input embeddings $E(x)$ is computed:

$$G = \nabla_{E(x)} L_{\text{clean}} \tag{2}$$

A **perturbation** R is then calculated by scaling the gradient, ensuring it is norm-bounded:

$$R = \epsilon \frac{G}{\|G\|_F} \tag{3}$$

where ϵ controls the magnitude of the perturbation and $\|G\|_F$ is the Frobenius norm of the gradient.

The **adversarial embeddings** $E_{\text{adv}}(x)$ are generated by adding the perturbation to the original embeddings:

$$E_{\text{adv}}(x) = E(x) + R \tag{4}$$

The **adversarial loss** L_{adv} is then computed using the adversarial embeddings:

$$L_{\text{adv}} = -\log p_\theta(y|x; E_{\text{adv}}(x)) \tag{5}$$

Finally, the total objective function L is a weighted sum of the clean loss and the adversarial loss, with λ controlling the balance between the two:

$$\boxed{\mathcal{L} \;=\; \mathcal{L}_{\text{clean}} \;+\; \lambda\, \mathcal{L}_{\text{adv}}} \tag{6}$$

where λ is a hyperparameter that determines the importance of adversarial training.

Equations 1--6 describe the objective employed in TACT using FGM. The clean loss L_{clean} is computed from the standard cross-entropy on clean data. The gradient of the clean loss is used to generate adversarial examples, and the adversarial loss L_{adv} is computed using the perturbed embeddings. The total loss L is a weighted sum of the clean and adversarial losses, with λ controlling the trade-off between them.

4.2 LLRD

LLRD (Ishii and Sato, 2017) makes transformer fine-tuning more stable by giving lower layers smaller learning rates and upper layers larger rates. Parameters are organized by depth, such as embeddings, encoder layers, and the classifier head. Each group is then optimized with its own learning-rate "bucket."

For an encoder with L layers indexed from bottom to top by $l \in \{0, \ldots, L-1\}$, the learning rate η_l for each layer is given by:

$$\eta_l = \eta_0 \alpha^{L-1-l}, \quad l = 0, \ldots, L-1 \tag{7}$$

where η_0 is the base learning rate applied to the top layer, $\alpha \in (0, 1)$ is the decay factor, and L is the total number of layers. Equation 7 ensures that the learning rate decreases progressively from the top layers to the bottom layers, with the decay factor α controlling the rate at which this reduction occurs.

For the embedding block, the learning rate is calculated separately:

$$\eta_{\text{emb}} = \eta_0 \alpha^L \tag{8}$$

This learning rate η_{emb} applies to the embedding layer, which is smaller than the learning rates of the upper layers, as it follows the same decay pattern.

4.2.1 AdamW with group-wise decay and LLRD

Let $\{\mathcal{G}_l\}$ be parameter groups aligned with depth l (plus an embedding group), and let $\mathbf{1}_{\text{decay}}(w) \in \{0, 1\}$ mask weight decay (e.g., $\mathbf{1}_{\text{decay}}(w)$=0 for biases and LayerNorm scales). The optimization objective is

$$\min_\theta \;\; E\big[\mathcal{L}(\theta)\big] \; + \sum_l \lambda_l \sum_{w \in \mathcal{G}_l} \mathbf{1}_{\text{decay}}(w)\, \|w\|_2^2 \tag{9}$$

with per-group step sizes set by the LLRD schedule:

$$\eta(\mathcal{G}_l) \;=\; \eta_l, \qquad \eta(\text{emb}) \;=\; \eta_{\text{emb}} \tag{10}$$

The Equation 7,8 explain LLRD, where the learning rate for each encoder layer (l) goes down based on a decay factor (α) and the total number of layers (L). The embedding block's learning rate is set to (η_{emb}) and is also decreased based on (α^L). The AdamW optimizer only applies weight decay to some parameters, and the total optimization objective is the loss function ($\mathcal{L}(\theta)$) plus the weight decay regularization in Equation 9, and 10. The LLRD schedule sets the step sizes for each group of parameters.

4.3 Logits Generation and Ensemble Technique

Logits generation involves getting raw output scores from each model for each sample in the test or validation set. For each input x_i, each model m produces logits $z_m(x_i)$, which are used to make the final predictions shown in Equation 11.

$$z_m(x_i) = f_m(B_i), \quad \forall x_i \in D_{\text{test}} \tag{11}$$

where f_m represents the function (or model) m applied to the input batch B_i. The logits from the models are then aggregated using a weighted sum to form the ensemble logits:

$$z_{\text{ens}}(x_i) = \sum_{m \in M'} w_m z_m(x_i) \tag{12}$$

where M' denotes the subset of models used in the ensemble, and w_m is the learned weight for each model. The final prediction for each subtask is obtained by applying the **argmax** function to the ensembled logits:

$$\hat{y}_i = \text{argmax}(z_{\text{ens}}(x_i)) \tag{13}$$

Model	Hate-type without TACT+LLRD	Hate-type TACT+LLRD	To-whom without TACT+LLRD	To-whom TACT+LLRD	Hate-severity without TACT+LLRD	Hate-severity TACT+LLRD
BanglaBERT	0.7013	0.7265	0.6914	0.7277	0.7270	0.7431
MuRIL	0.6992	0.7179	0.7087	0.7143	0.7022	0.7305
mE5-base	0.7122	0.7220	0.6961	0.7105	0.7228	0.7303
Twitter XLM-R	0.6986	0.7100	0.7089	0.6917	0.7024	0.7232

Table 1: Performance comparison of our Models on F1 score

Ensembling $z_{\text{ens}}(x_i)$combines the logits from different model combinations. To get the final prediction for each subtask, use$\arg\max$ of the ensembled logits shows in Equation 12, 13.

4.4 Our Approach

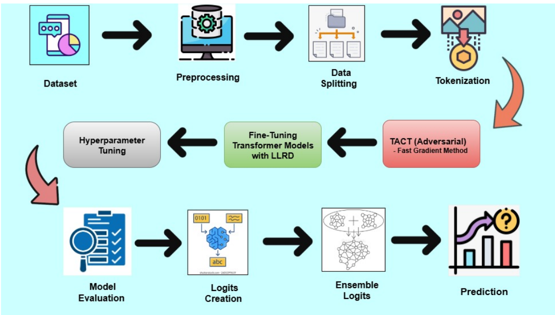


Figure 1: Workflow Diagram of our proposed methodology

Figure 1 illustrates our framework for multi-class Bengali hate speech detection. The first step is preprocessing (cleaning, normalizing, and splitting the dataset into groups), and then tokenization to ensure uniform input representation. We add TACT with FGM during training to improve accuracy and get better results. Furthermore, we use AdamW with LLRD to fine-tune BanglaBERT and other Transformer models (Multilingual-E5, MuRIL, Twitter XLM-R). In the end, we perform hyperparameter tuning, and the logits from each model are standardized and combined through an ensemble just by adding the logits, which makes the predictions more accurate and reliable. The algorithm of our whole process is shown in Appendix C.

5 Result Analysis

As defined in our methodology 4.4, **TACT with LLRD** has performed significantly well, outperforming benchmark results of separate models, as shown in Table 1. Ensembling different models' logits through aggregation, which were standardised, further improved the performance. This ensemble technique is showed in equation 12. Using fixed, uniform ensemble weights (i.e., $w_m = 1$ for all models) further improved our performance across **Subtasks 1A, 1B, and 1C**. The results are presented in Tables 9, 10, and 11, which summarize all ensemble combinations evaluated in our experiments. The combinations for which we got the peak scores in each task are shown in Table 2. However, we also performed other methods.
Instead of aggregating the results, we tried to use a neural network to learn the weights w_m. However, it could not achieve the peak accuracy as shown in Table 13. To address the dataset imbalance, we also experimented with a two-step classification approach. First, we trained a binary classifier to distinguish between None and Not None, achieving an accuracy of 0.7718 for this binary task. In the second step, only the instances classified as Not None were further assigned to the remaining classes. However, this pipeline resulted in an overall accuracy of only 0.7127.
Despite these approaches, we also tried to mitigate the imbalance of the dataset with other methods, which are shown in Table 12 with accuracy. However, none of these models could beat the superior result of our novel approach.

Sub task	Class	Ensemble Model	F1-score
1A	Hate-type	BanglaBERT MuRIL mE5-base	0.7362
1B	To-whom	BanglaBERT MuRIL mE5-base	0.7335
1C	Hate-type To-whom Hate-severity	BanglaBERT + MuRIL + mE5-base BanglaBERT + mE5-base BanglaBERT + MuRIL + mE5-base	0.7361

Table 2: F1-scores of the best-performing ensemble combinations for all subtasks.

Our initial pool included seven pretrained encoder models chosen for their ability to capture the nuances and intricacies of Bengali. We found 4 models which were consistent among the tasks shown in Table 8. Although Twitter XLM-R had a lower accuracy than some of the models, as shown in the table, it was chosen because of its superior performance in other tasks.

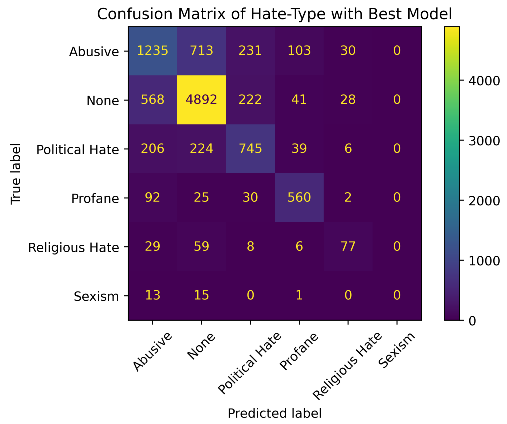


Figure 2: Confusion Matrix for Subtask 1A (Hate Type) using the Best-Performing Ensemble Model

5.1 Error Analysis

In Figure 2, we can see the classification report of the ensemble model (BanglaBERT + MuRIL + mE5-base), which achieved the top performance score in the leaderboard for the hate-type class. The ensemble models performed well for the None (85.06%) and Profane (78.98%) classes, but not very well for the Political Hate (61.06%) and Abusive (58.69%) classes. Also, the results for Religious Hate and Sexism were not consistent; for example, there were no correct predictions for Sexism. This difference is mostly because the datasets were not balanced, since these classes had a lot fewer samples. In fact, a direct comparison between class frequency and per-class F1-score shows that classes with more samples achieve higher performance, while underrepresented classes—such as Religious Hate, and Sexism—consistently perform worse. Thus, data imbalance can largely be attributed to this underperformance. Similar reasoning can be made for the other classes, and this imbalance of the dataset will be the reason for the inferior performance for the other classes.
However, data imbalance is not the sole reason for poor performance; the dataset contains samples that may fit multiple classes due to intertwined semantics. Certain Bengali sentences have overlapping meanings, causing ambiguity that remains unresolved even through human evaluation, as shown in Table 14.

6 Conclusion

This research developed a comprehensive framework for detecting Bangla hate speech by employing TACT with LLRD on transformer models, which was augmented by logits ensembling. The approach achieved the highest score, ranking first in subtask A, second in subtask B, and third in subtask C. Despite existing challenges such as imbalanced datasets and linguistic disparities, the suggested technique represents a commendable initial step towards enhancing hate speech identification in resource-scarce languages. It also provides valuable insights for other multilingual initiatives, paving the way for future works, making languages accessible and communication easier.

Limitations

There were several limitations in our work. The dataset, which was provided, had a small size and was highly imbalanced, as shown in the Figure 3. This imbalance had a lasting effect on our results, and despite trying a lot of approaches, the imbalance was noteworthy. Secondly, we chose 7 initial models based on their applicability in Bengali. These models were carefully curated. However, there may be further models that can be explored. Thirdly, the dataset had some mislabeled data, as shown in the error analysis, which had a detrimental effect on the accuracy.

References

Abdullah Al Maruf, Ahmad Jainul Abidin, Md Mahmudul Haque, Zakaria Masud Jiyad, Aditi Golder, Raaid Alubady, and Zeyar Aung. 2024. Hate speech detection in the bengali language: a comprehensive survey. *Journal of Big Data*, 11(1):97.

Fatimah Alkomah and Xiaogang Ma. 2022. A literature review of textual hate speech detection methods and datasets. *Information*, 13(6):273.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, and Md Farhad Alam. 2024. Banth: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing*

(BLP-2025), India. Association for Computational Linguistics.

Qiushi Huang, Tom Ko, H Lilian Tang, Xubo Liu, and Bo Wu. 2021. Token-level supervised contrastive learning for punctuation restoration. *arXiv preprint arXiv:2107.09099*.

Masato Ishii and Atsushi Sato. 2017. Layer-wise weight decay for deep neural networks. In *Pacific-Rim Symposium on Image and Video Technology*, pages 276--289. Springer.

Md Samiul Islam, Rifat Nawaz, Jannatul Ferdous Salma, Md Kamrul Islam, Mahabubur Rahman, Sapayev Valisher Odilbek, and Muhabbat Jumaniyozova. 2025. Leveraging banglabert: A transformer-based approach for multiclass hate speech detection in bangla social media. In *2025 4th International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, pages 487--492. IEEE.

Prashant Kapil and Asif Ekbal. 2025. A transformer based multi task learning approach to multimodal hate speech detection. *Natural Language Processing Journal*, 11:100133.

Md Rezaul Karim, Sumon Kanti Dey, Tanhim Islam, Sagor Sarker, Mehadi Hasan Menon, Kabir Hossain, Md Azam Hossain, and Stefan Decker. 2021. Deephateexplainer: Explainable hate speech detection in under-resourced bengali language. In *2021 IEEE 8th international conference on data science and advanced analytics (DSAA)*, pages 1--10. IEEE.

Snaholata Mondal, Md Samiul Alom, Md Mahbub Alum, and Kazi Lamia Sinja Sunjida. 2025. Multiclass detection of misogynistic bangla text from youtube using transformer-based models. In *2025 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1--6. IEEE.

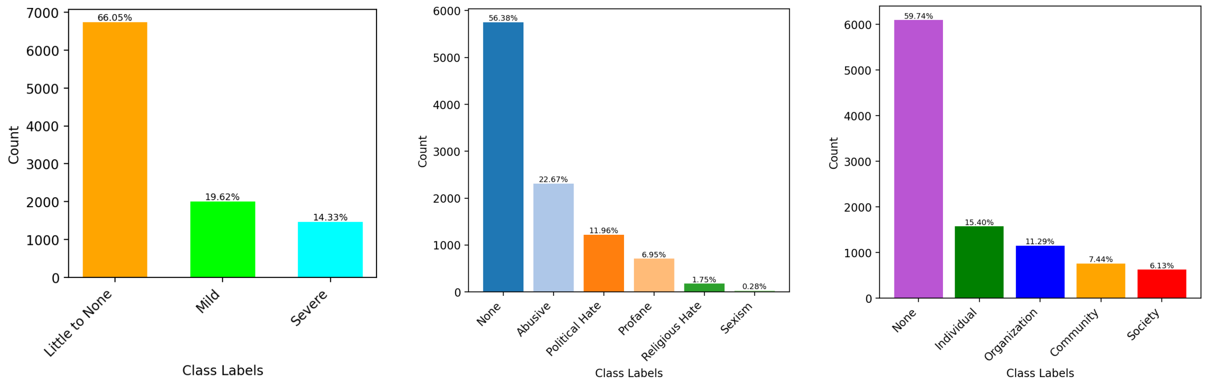
Md Nishat Raihan, Dhiman Goswami, Sadiya Sayara Chowdhury Puspo, and Marcos Zampieri. 2023. nlpbdpatriots at blp-2023 task 1: A two-step classification for violence inciting text detection in bangla. In *The First Workshop on Bangla Language Processing (BLP-2023)*, page 179.

Sourav Saha, Jahedul Alam Junaed, Maryam Saleki, Arnab Sen Sharma, Mohammad Rashidujjaman Rifat, Mohamed Rahouti, Syed Ishtiaque Ahmed, Nabeel Mohammed, and Mohammad Ruhul Amin. 2023. Vio-lens: A novel dataset of annotated social network posts leading to different forms of communal violence and its evaluation. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 72--84.

Yixuan Su, Fangyu Liu, Zaiqiao Meng, Tian Lan, Lei Shu, Ehsan Shareghi, and Nigel Collier. 2021. Tacl: Improving bert pre-training with token-aware contrastive learning. *arXiv preprint arXiv:2111.04198*.

Farhana Hossain Swarnali, Jannatim MaishaL, Muhammad Azmain Mahtab, M Saymon Islam Iftikar, and Faisal Muhammad Shah. 2024. Bengali multi-class text classification via enhanced contrastive learning techniques. In *2024 27th International Conference on Computer and Information Technology (ICCIT)*, pages 1576--1581. IEEE.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Cagrı Cöltekin. 2023. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020).



(a) Hate Severity Distribution

(b) Hate Type Distribution

(c) Target Distribution

Figure 3: Dataset class distributions across severity, type, and target dimensions.

A Dataset Samples

The given dataset is for hate speech in Bangla across severity, type and target. As shown in Figure 3, severity is biased towards less harmful speech, and type is towards abuse and religious hate. The target is towards individuals more than organizations or communities.

id	text	label
432313	আসলে মহান নেতা এটাই তার পরিচয়	None
359516	ইরান ধ্বংশ হউক	Abusive
578332	আলহামদুলিল্লাহ দেশ এগিয়ে যাচ্ছে বিএনপি জামাতীদের জুতা পেটা করতে হবে	Political Hate
404893	সময় টিভি একটা জাউড়া মিডিয়া মিথ্যা তথ্য প্রচার করে বেড়ায়	Profane
764029	ইহুদি নাসাদের শিক্ষা মুসলমানদের জন্য হারাম	Religious Hate
639002	স্লোগানে আমি প্রথমে যৌন নেত্রীর আগমন শুনছি	Sexism

Table 3: Examples of hate type dataset samples.

id	text	label
165894	হেন কাপ পুলিশের মারে অন্যরা তাহলে পলিশের কি হবে বিচার হবে কি	None
587800	ওনি আমার বালের ওলি বালের ভান্ডারী কুত্তার বাচ্চারা সব ভন্ড	Individual
241030	ভারতীয় দালাল সময় টিভিকে বয়কট করুন	Organization
124999	আল্লাহ্ এসব জানোয়ারদের শেষ করে দাও	Community
12764	ইজরায়েলের বিচার হওয়া উচিৎ	Society

Table 4: Examples of to whom dataset samples.

id	text	label
165894	হেন কাপ পুলিশের মাঝে অন্যরা থাকলে পুলিশের কি হবে বিচার হবে কি	Little to None
814896	হালার এই দেশে বড় আইনের ফুজিটিভদের বাসা বাড়িতে দিয়ে দেয় খালাসন মাজার জন্য	Mild
124999	আল্লাহ এইসব জানোয়ারদের শেষ করে দাও	Severe

Table 5: Examples of hate severity dataset samples.

B Baseline & Observations

There were several noteworthy observations. We found that during validation, the model chosen with the best validation accuracy resulted in a better overall model. In the ensemble method, we found that logits, when standardized, had a better impact on the score. Also, all the approaches that we tried for a single model had a poorer result than Banglabert, which was consistently the best model throughout the tasks.

The organisers have also provided baseline results for this task on both the Dev-Test and Test Datasets. Three different models were used: the Random Baseline, Majority Baseline, and the n-gram Baseline. As shown in the Table 7, our model did much better than all of the baselines on both the Test and Dev sets. On the Test set, it got micro-F1 scores of 0.7362, 0.7335, and 0.7361 across Subtasks 1A, 1B, and 1C. On the Dev set, it got even better scores of 0.7579, 0.7531, and 0.7558.

C Algorithm

Algorithm 1: Multitask Bangla Hate Speech Detection with FGM (TACT) and LLRD

```
Input: $\mathcal{D} = \{(x_i, y_i^{(A)}, y_i^{(B)}, y_i^{(C)})\}_{i=1}^{N}$; model set $\mathcal{M}$; FGM radius $\epsilon$; mix
       weight $\lambda_{\text{adv}}$; LLRD decay $\alpha \in (0, 1)$; train ratio $r{=}0.7$
Output: Prediction labels for 1A (type), 1B (severity), 1C (target)
Preprocess & split: Clean/normalize texts; Split $\mathcal{D} \to \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$ with $|\mathcal{D}_{\text{train}}|/N \approx r$.;
foreach model $m \in \mathcal{M}$ do
    Tokenize with $m$'s tokenizer ;
    Build AdamW with LLRD parameter groups: for encoder layer $l{=}0 \dots L{-}1$, set $\eta_l \leftarrow \eta_0\, \alpha^{\,L-1-l}$;;
    embeddings $\eta_{\text{emb}} \leftarrow \eta_0\, \alpha^{L}$; (no-decay for biases/LN).;
    for epoch $= 1 \dots E$ do
        foreach mini-batch $\mathcal{B} \subset \mathcal{D}_{train}$ do
            // Clean forward & loss
            Get logits $z = f_\theta(\mathcal{B})$; $\mathcal{L}_{\text{clean}} = \mathrm{CE}(\mathrm{softmax}(z), y)$.;
            // FGM perturbation (TACT)
            $G = \nabla_E \mathcal{L}_{\text{clean}}$;;
            $R = \epsilon\, G / \|G\|_F$;;
            set $E_{\text{adv}} = E + R$.;
            Get $z_{\text{adv}} = f_\theta(\mathcal{B}; E_{\text{adv}})$; $\mathcal{L}_{\text{adv}} = \mathrm{CE}(\mathrm{softmax}(z_{\text{adv}}), y)$.;
            // Total loss & update
            $\mathcal{L} = \mathcal{L}_{\text{clean}} + \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}$; update $\theta$ with LLRD rates.;
        Validate on $\mathcal{D}_{\text{val}}$; keep best checkpoint.;
    Generate/store per-subtask logits on dev/test.;
Ensemble: For each sample, combine logits across chosen $\mathcal{M}' \subseteq \mathcal{M}$.
```

This algorithm depicts the whole approach where we fine-tune transformer models for Bangla hate speech detection using TACT and LLRD. In each epoch tokenized text is perturbed and passed through the model to compute the clean and adversarial loss, and then the parameters are updated. Finally logits are generated for each subtask (hate type, severity, target), and an ensemble is applied for prediction.

D Experimental Setup and Hyperparameter

We fine-tuned transformer-based encoders like BanglaBERT, MuRIL, multilingual E5-base, and Twitter XLM-R on the Bangla hate speech dataset for three different tasks. We tokenized the preprocessed text and put it into models that had been trained with AdamW and Layer-wise Learning Rate Decay (LLRD). Token-Aware Adversarial Contrastive Training (TACT) was employed with minor adjustments at the embedding level to enhance the system's robustness.

Hyperparameter	**Value**
Number of epochs	5
Train batch size (per device)	16
Eval batch size (per device)	16
Learning rate	2e-5
Weight decay	0.01
Warmup ratio	0.1

Table 6: Selected hyperparameters used for model training.

We trained for five epochs with a batch size of 16, a learning rate of 2e-5, a weight decay of 0.01, and a warm-up ratio of 0.1 as shown in Table 6. To make sure that models were stable, hyperparameters were tuned within small ranges. Final predictions were made by combining the logits of different models based on how well they did on the development set.

Subtask 1A Model	**micro-F1**	**Subtask 1B Model**	**micro-F1**	**Subtask 1C Model**	**weighted micro-F1**
Random Baseline	0.1638	Random Baseline	0.2043	Random Baseline	0.2304
Majority Baseline	0.5638	Majority Baseline	0.5974	Majority Baseline	0.6072
n-gram Baseline	0.6020	n-gram Baseline	0.6209	n-gram Baseline	0.6305
Our Model (TestSet)	**0.7362**	Our Model (TestSet)	**0.7335**	Our Model (TestSet)	**0.7361**
Our Model (DevSet)	**0..7579**	Our Model (DevSet)	**0.7531**	Our Model (DevSet)	**0.7558**

Table 7: Comparison with the baseline

Model Name	**Model name (Short)**	**F1-score (Dev Test)**
csebuetnlp/banglabert	BanglaBERT	**0.7389**
google/muril-base-cased	MuRIL	0.7281
intfloat/multilingual-e5-base	mE5-base	0.7253
cardiffnlp/twitter-XLM-R-base-sentiment	Twitter XLM-R	0.7134
sagorsarker/bangla-bert-base	sagorsarker_bert	0.7054
distilbert-base-multilingual-cased	distilbert	0.7166
FacebookAI/roberta-base	xlm-roberta	0.7209

Table 8: Performance of different pretrained models on the development test set.

Task Name	**Ensemble Model Combination**	**F1-Score**
Subtask 1A (Hate Type)	BanglaBERT	0.7265
	MuRIL	0.7179
	mE5-base	0.7220
	Twitter XLM-R	0.7100
	BanglaBERT + MuRIL	0.7336
	BanglaBERT + mE5-base	0.7358
	BanglaBERT + Twitter XLM-R	0.7299
	MuRIL + mE5-base	0.7277
	MuRIL + Twitter XLM-R	0.7246
	mE5-base + Twitter XLM-R	0.7209
	BanglaBERT + MuRIL + mE5-base	**0.7361**
	BanglaBERT + MuRIL + Twitter XLM-R	0.7323
	BanglaBERT + mE5-base + Twitter XLM-R	0.7331
	MuRIL + mE5-base + Twitter XLM-R	0.7292
	BanglaBERT + MuRIL + mE5-base + Twitter XLM-R	0.7327

Table 9: F1-scores of different ensemble model combinations for Subtask 1A (Hate Type). The best-performing score is highlighted in bold.

Task Name	**Ensemble Model Combination**	**F1-Score**
Subtask 1B (To-whom)	BanglaBERT	0.7277
	MuRIL	0.7143
	mE5-base	0.7105
	Twitter XLM-R	0.6917
	BanglaBERT + MuRIL	0.7314
	BanglaBERT + mE5-base	0.7319
	BanglaBERT + Twitter XLM-R	0.7312
	MuRIL + mE5-base	0.7312
	MuRIL + Twitter XLM-R	0.7255
	mE5-base + Twitter XLM-R	0.7193
	BanglaBERT + MuRIL + mE5-base	**0.7335**
	BanglaBERT + MuRIL + Twitter XLM-R	0.7334
	BanglaBERT + mE5-base + Twitter XLM-R	0.7299
	MuRIL + mE5-base + Twitter XLM-R	0.7268
	BanglaBERT + MuRIL + mE5-base + Twitter XLM-R	0.7330

Table 10: F1-scores of different ensemble model combinations for Subtask 1B (To-whom). The best-performing score is highlighted in bold.

Task Name	Ensemble Model Combination (Hate Type)	Ensemble Model Combination (To-Whom)	Ensemble Model Combination (Hate Severity)	F1-Score
Subtask 1C (Hate-Type, To-Whom, Hate-Severity)	BanglaBERT + Multilingual	BanglaBERT + MuRIL	BanglaBERT + Multilingual	0.7345
	BanglaBERT + MuRIL + Multilingual	BanglaBERT + Multilingual	BanglaBERT + MuRIL + Multilingual	0.7342
	BanglaBERT + MuRIL	BanglaBERT + MuRIL + Multilingual	BanglaBERT + Multilingual	0.7356
	BanglaBERT + Multilingual	BanglaBERT + Multilingual	BanglaBERT + Multilingual	0.7353
	BanglaBERT + MuRIL + Multilingual	BanglaBERT + Multilingual	BanglaBERT + MuRIL + Multilingual	**0.7361**
	BanglaBERT + MuRIL	BanglaBERT + MuRIL + Multilingual	BanglaBERT + MuRIL + Multilingual	0.7358
	BanglaBERT + Multilingual	BanglaBERT + Multilingual	BanglaBERT + Multilingual	0.7341
	BanglaBERT + MuRIL + Multilingual	BanglaBERT + MuRIL	BanglaBERT + MuRIL + Multilingual	0.7328
	BanglaBERT + MuRIL	BanglaBERT + Multilingual	BanglaBERT + Multilingual	0.7352
	BanglaBERT + Multilingual	BanglaBERT + MuRIL + Multilingual	BanglaBERT + Multilingual	0.7349
	BanglaBERT + MuRIL + Multilingual	BanglaBERT + Multilingual	BanglaBERT + MuRIL + Multilingual	0.7357
	BanglaBERT + Multilingual	BanglaBERT + MuRIL	BanglaBERT + Multilingual	0.7354

Table 11: F1-scores of different ensemble model combinations for Subtask 1C (Hate-Type, To-Whom, Hate-Severity). The best-performing score is highlighted in bold.

Serial No	Model	Validation Accuracy
1	Binary classification	0.7127
2	LSTM with Banglabert logits	0.5820
3	Class-Weighted Loss	0.6644
4	Class-Weighted Loss with TACT	0.6835
5	Random Over-Sampling	0.7026
6	Focal Loss	0.6696
7	Banglabert	0.7229

Table 12: Validation accuracy of other approaches.

Serial No	Model	F1-score (Validation Test)
1	Linear Neural Network with 3 logits	0.7245
2	Multilayer perceptron with 3 logits	0.7258
3	Linear Neural Network with 2 logits	0.7365
4	Multilayer perceptron with 2 logits	0.7372
5	Our best ensemble model	**0.7580**

Table 13: F1-scores of different ensemble approaches on validation test.

ID	Text	Label	Correct Label
891820	আওয়ামী লীগের বাকশালী রাজনীতি ঢুকে গেছে ক্রিকেট টিমে	Abusive	Political Hate
272543	চাচি তুমি ভোট চোর	Political Hate	Abusive
269938	কি উন্নয়ন আলু ৭০ পেয়াজ ১৪০ চাল ৭০ হেইয়ো উন্নয়ন	Political Hate	None
428911	এই সব বিক্ষোভ আন্দোলন ইসরাইল ভয় পায় না	Religious Hate	Political Hate
813236	তীব্র গরম জাহান্নামের নিঃশ্বাস	Religious Hate	None
165501	মেয়েদের কোথাও নিরাপদ নেই	Sexism	None
514297	এ শলায় মেয়ে লোকের দালাল শাকিব খান	None	Abusive

Table 14: Original and proposed classes for possible misclassified samples in Subtask A (Hate-type).

CUET_Sntx_Srfrs at BLP-2025 Task 1: Combining Hierarchical Classification and Ensemble Learning for Bengali Hate Speech Detection

Hafsa Hoque Tripty, Laiba Tabassum, Hasan Mesbaul Ali Taher, Kawsar Ahmed, Mohammed Moshiul Hoque
Department of Computer Science and Engineering,
Chittagong University of Engineering & Technology, Chittagong 4349, Bangladesh
{u2204108,u2204077,u1804038,u1804017}@student.cuet.ac.bd
moshiul_240@cuet.ac.bd

Abstract

Detecting hate speech in Bengali social media content presents considerable challenges, primarily due to the prevalence of informal language and the limited availability of annotated datasets. This study investigates the identification of hate speech in Bengali YouTube comments, focusing on classifying the type, severity, and target group. Multiple machine learning baselines and voting ensemble techniques are evaluated to address these tasks. The methodology involves text preprocessing, feature extraction using TF-IDF and Count vectors, and aggregating predictions from several models. Hierarchical classification with TF-IDF features and majority voting improves the detection of less frequent hate speech categories while maintaining robust overall performance, resulting in an 18^{th} place ranking and a micro F1 score of 68.42%. Furthermore, ablation studies assess the impact of preprocessing steps and n-gram selection, providing reproducible baselines for Bengali hate speech detection. All codes and resources are publicly available at GitHub[1].

1 Introduction

Hate speech detection refers to the automated identification of language that targets individuals or groups with hostility or discrimination based on attributes such as gender, religion, or political affiliation. Detecting hate speech is essential for maintaining safe online environments and is widely used in content moderation, policy enforcement, and research on social dynamics. Detecting hate speech is challenging, especially in less-resourced languages like Bengali, where annotated datasets and robust methods are limited despite growing online activity. Recently, several methods have shown promise for low-resource hate speech detection (Sanoussi et al., 2022; Alaoui et al., 2022). However, most of these focused on binary classification of hate speech, overlooking hate type, severity, or target group (Chiril et al., 2019; Alam et al., 2024; Hossain et al., 2023). This restricts their ability to capture the multi-dimensional nature of hate speech in Bengali (Das et al., 2022). The Bengali multi-task hate speech identification shared task (Hasan et al., 2025b) addresses this with three subtasks: (1A) hate type, (1B) target group, and (1C) multi-task classification of type, severity, and target.

This study investigates machine learning methods for multi-task Bengali hate speech detection in the context of significant data imbalance. Our approach integrates preprocessing, n-gram vectorization, multiple machine learning models, hierarchical classification, and ensemble learning. The main contributions are:

- Developed a framework for Bengali hate speech detection that uses hierarchical classification and majority voting within machine learning ensembles.
- Conducted a systematic evaluation of 16 models across three subtasks, including comprehensive error analysis to assess model effectiveness.

2 Related Work

Hate speech detection has been widely explored in high-resource languages such as English, Arabic, and other European languages. Early studies primarily used machine learning (ML) with handcrafted features. Sanoussi et al. (2022) reported

[1] https://github.com/Hasan-Mesbaul-Ali-Taher/BLP_25_Task_1

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 523–530
December 23, 2025 ©2025 Association for Computational Linguistics

an accuracy of 95.45% using an SVM classifier, while Alaoui et al. (2022) achieved an accuracy value of 87.23% in the first dataset, and the second dataset attained a value of 93.06%. Al-Hassan and Al-Dossari (2022) explored various ML and DL models, and they achieved the maximum F1 score of 73% on Arabic tweets with the ensemble method (CNN+LSTM). Saleh et al. (2023) utilized BiLSTM for hate speech detection in English and achieved an F1 score of 93%. Khan et al. (2022a) introduced BiCHAT, which leverages BiLSTM and deep CNNs, achieving an F1 score of 0.84 on a Twitter dataset and surpassing benchmarks (Khan et al., 2022b). In contrast, Research on low-resource languages has received relatively limited attention in hate speech detection due to data scarcity and linguistic diversity. Bade et al. (2024) focused on Dravidian languages, employing a Random Forest (RF) model with TF-IDF features and obtained an F1 score of 0.492. (Ibrohim and Budi, 2019) utilized RF+DT for multi-label hate speech detection in Indonesian languages, achieving an accuracy of 77.36%. At the same time, Nasir et al. (2024) proposed a two-level LR technique for Roman-Urdu code-mixed data, which gained 87%accuracy. Bilal et al. (2023) further applied the BERT model to detect hate speech from Roman-Urdu code-mixed text and achieved 97.89% accuracy.

Bengali, as a low-resource language in NLP, lacks a standard benchmark dataset for this task, making consistent evaluation difficult. Due to these constraints, very few research activities have been carried out in this area of Bangla Language Processing (BLP), which are mostly related to hate pr offensive content from social platforms. Momin and Sarker (2025) explored various machine learning and deep learning techniques for hate speech detection and achieved 95% accuracy using a Bi-LSTM with FastText embeddings, while Saha et al. (2024) employed a hybrid BERT-CNN model and achieved an F1 score of 94.44%. Several transformer-based models and LLMs have been explored by Haider et al. (2025) for hate speech detection in transliterated Bengali. This work achieved the highest F1 score of 77.36 with the mBERT model. A recent study utilized BanglaBERT and achieved an accuracy of 74%. Another study employed a hierarchical BERT model (Das et al., 2023) and obtained an F1 score of 0.73797. Faruqe et al. (2023) utilized GRU techniques and gained the highest accuracy of 98.87%. Most previous studies on Bengali hate speech detection have relied on small, imbalanced, and domain-specific datasets, making robust generalization difficult. These works primarily target hate speech in social media posts, neglecting other critical contexts, such as online news comments and discussion forums. To the best of our knowledge, there is still no unified approach for Bengali hate speech detection. This work aims to fill this gap by introducing a more comprehensive approach to detecting hate speech in Bengali.

3 Task and Dataset Description

The Bangla Multi-task hate speech identification shared task (Hasan et al., 2025b) comprises three interrelated subtasks for comprehensive analysis of hate speech on Bangla social media.

- **Subtask 1A:** Classify texts into *Abusive (Ab), Sexism (Sism), Religious Hate (RHate), Political Hate (PHate), Profane (Pfane)*, or *None (Non).*

- **Subtask 1B:** Identify the target as *Individual (In), Organization (Org), Community (Co), Society (So)*, or *None (Non).*

- **Subtask 1C:** Joint classification of hate type, severity (*Little to None (Non), Mild (Mild), Severe (Severe)*), and target group.

3.1 Datasets

The **BanglaMultiHate** corpus (Hasan et al., 2025a) comprises public YouTube comments on politics, sports, and international topics, preprocessed to remove non-Bangla literals and punctuation. Tables 1–3 summarize the distribution of classes across the training, development, and test splits for each subtask. These distributions highlight the imbalance among labels, which poses challenges for model training and evaluation.

Each of the three datasets exhibits strong class imbalance. For Subtask 1A, the *None* class accounts for more than 56% of the training set, whereas minority labels such as *Sexism* (0.34%)

Label	Train	Dev	Test
None	19,954	1,451	5,751
Abusive	8,212	564	2,312
Political Hate	4,227	291	1,220
Profane	2,331	157	709
Religious Hate	676	38	179
Sexism	122	11	29
Total	**35,522**	**2,512**	**10,200**

Table 1: Dataset statistics for hate type.

Label	Train	Dev	Test
None	21,190	1,536	6,093
Individuals	5,646	364	1,571
Organizations	3,846	292	1,152
Communities	2,635	179	759
Society	2,205	141	625
Total	**35,522**	**2,512**	**10,200**

Table 2: Dataset statistics for hate target.

Label	Train	Dev	Test
Little to None	23,489	1,703	6,737
Mild	6,853	483	2,001
Severe	5,180	326	1,462
Total	**35,522**	**2,512**	**10,200**

Table 3: Dataset statistics for hate severity.

and *Religious Hate* (1.9%) are extremely underrepresented. Subtask 1B follows a similar trend, with *None* comprising nearly 60% of all instances, while *Community* and *Society* each account for less than 7%. In Subtask 1C, *Little to None* dominates two-thirds of the dataset, and the combined proportion of *Mild* and *Severe* remains below 34%.

4 Methodology

This work leverages classical machine learning to establish strong baselines for Bangla hate speech detection under resource constraints, using careful preprocessing, hierarchical classification, and ensemble learning. Figure 1 illustrates the abstract process of hate speech detection.

- **Preprocessing:** To effectively manage noisy Bangla social media text, we performed number removal, stopword elimination using a curated Bangla list, and lemmatization to reduce words to their base forms. The stopword lists and the lemmatizer used are mentioned in the GitHub link. The impact of these preprocessing steps is analyzed in the Ablation Study (Section 5.1).

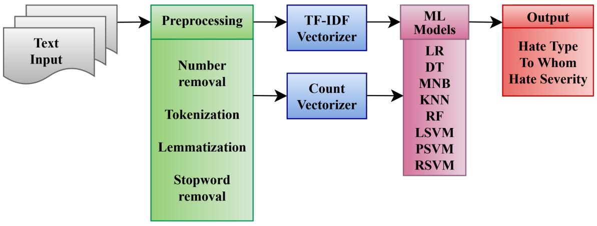


Figure 1: Abstract process of hate speech classification.

- **Tokenization and Vectorization:** Texts were tokenized using whitespace, and features were encoded with Unigram–Bigram–Trigram Count and TF-IDF vectorizers (min_df = 5). The effect of varying n-gram configurations is examined in the Ablation Study (Section 5.1).

- **Classification Models:** Logistic Regression, Decision Tree, Multinomial Naive Bayes, Support Vector Classifier (linear, polynomial, radial), Random Forest, and k-Nearest Neighbors were paired with both vectorizers, forming 16 model–vectorizer combinations.

- **Simple vs. Hierarchical Classification:** In standard classification tasks, models directly predict the target labels. In contrast, hierarchical classification (Figure 2) involves an initial separation between *None* and *Hate* (or *Little to None* and *Hate*). Only Hate instances are subsequently processed by a second classifier, which determines the specific hate category. This method facilitates more detailed identification and categorization of hate-related content. The first model produces $f_1(x) \in \{\text{None}, \text{Hate}\}$, and if $f_1(x) = \text{Hate}$, the second model assigns $f_2(x)$ from the hate-related classes. The final prediction is determined as follows:

$$F(x) = \begin{cases} \text{None}, & \text{if } f_1(x) = \text{None}, \\ f_2(x), & \text{if } f_1(x) = \text{Hate}. \end{cases}$$

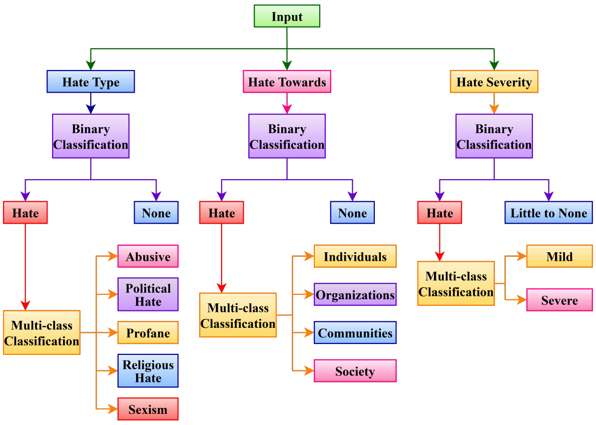


Figure 2: Hierarchical strategy for hate speech classification.

- **Ensemble:** A majority-voting ensemble was applied using the top three models (LR-Count-Hierarchical, SVC-lin-Count-Hierarchical, SVC-rbf-Count-Hierarchical). For each instance, the label agreed upon by at least two models was chosen; if all disagreed, the best-performing model's prediction was used (Algorithm 1).

Algorithm 1 Ensemble Voting for Hate Classification

```
Input: Predictions $P_1, P_2, P_3$ from top-3 models; best-performing model $M^*$
Output: Final predicted label $L$
for each instance $x$ do
    Collect predictions $\{p_1, p_2, p_3\}$
    if two or more predictions agree then
        Assign $L$ to the majority label
    else
        Assign $L$ to the prediction of $M^*$
    end if
end for
Return all $L$
```

4.1 Hyperparameter Settings

Table 4 lists the key hyperparameters used for each machine learning model in our experiments. These settings were selected based on preliminary tuning to balance performance and computational efficiency across the three subtasks.

Model	Key Hyperparameters
LR	C = 1, solver = liblinear
DT	criterion = entropy, min_samples_leaf = 2, random_state = 0
MNB	alpha = 0.5
KNN	n_neighbors = 7, weights = distance
RF	class_weight = balanced, criterion = entropy, max_features = log2, min_samples_leaf = 2, n_estimators = 10, random_state = 0
SVC-l	C = 0.2, kernel = linear, probability = True
SVC-p	C = 10, kernel = poly, gamma = auto
SVC-r	C = 1000, kernel = rbf, gamma = 0.00015, probability = True

Table 4: Hyperparameters used for classification models.

5 Results

All experiments were implemented in Python using scikit-learn and executed on Google Colab using a standard CPU, ensuring reproducibility across all subtasks. The overall pipeline was designed as a modular framework that covers preprocessing, feature extraction, training, and evaluation, enabling consistent comparisons of models across identical settings. Table 5 summarizes the performance of all model-vectorizer combinations under both simple and hierarchical settings.

Model-Vectorizer	F1-micro (Simple)	F1-micro (Hierarchical)
LR-Count	68.12	**68.29**
LR-TF-IDF	67.90	68.10
DT-Count	62.80	63.00
DT-TF-IDF	59.00	59.20
MNB-Count	64.10	64.30
MNB-TF-IDF	66.30	66.50
KNN-Count	63.10	63.30
KNN-TF-IDF	61.60	61.80
RF-Count	56.60	56.80
RF-TF-IDF	55.60	55.80
SVC-lin-Count	68.10	**68.32**
SVC-lin-TF-IDF	66.20	66.40
SVC-poly-Count	60.60	60.80
SVC-poly-TF-IDF	60.60	60.80
SVC-rbf-Count	68.10	**68.30**
SVC-rbf-TF-IDF	67.00	67.20
Majority Voting	–	**68.42**

Table 5: Performance of various models.

The majority voting ensemble achieved the highest F1-micro (**68.42%**), demonstrating robustness across setups. Logistic Regression and SVM consistently outperformed tree- and distance-based models, while hierarchical classification improved detection of minority classes by separating hate from non-hate content. TF-IDF favored linear models, whereas CountVectorizer slightly benefited tree-based methods. Overall, hierarchical classification with TF-IDF and ensemble voting provided the best balance between minority-class sensitivity and global accuracy.

5.1 Ablation Study

This subsection presents an ablation study analyzing the impact of preprocessing techniques and n-gram choices on model performance, along with token statistics at each preprocessing stage.

Impact of Text Preprocessing: We evaluated different preprocessing pipelines on Task 1C using Logistic Regression with trigram features (simple classification). The preprocessing variants are summarized in Table 6.

Preprocessing	**F1-Micro (%)**
text	66.60
nr_text	66.64
l_nr_text	67.31
sr_nr_text	67.12
sr_l_nr_text	67.90

Table 6: Effect of preprocessing on F1-micro. Preprocessing variations: `text` = raw text, `nr_text` = numbers removed, `l_nr_text` = lemmatized + numbers removed, `sr_nr_text` = stopwords removed + numbers removed, `sr_l_nr_text` = stopwords removed + lemmatized + numbers removed.

Impact of N-gram Choice: We tested unigram, unigram+bigram, and unigram+bigram+trigram TF-IDF vectorization with the best preprocessing pipeline on Task 1C (`sr_l_nr_text`) using Logistic Regression in the simple classification setting. The effect of n-gram choice on model performance is shown in Table 7.

Token Counts at Each Preprocessing Stage: Table 8 summarizes total and unique token counts for train, dev, and test sets in Task 1C.

N-gram	**F1-Micro (%)**
Unigram	67.82
Unigram+Bigram	67.75
Unigram+Bigram+Trigram	67.90

Table 7: Effect of n-gram choice on F1-micro.

Dataset	**Preprocessing**	**Total Tokens**	**Unique Tokens**
train_1C_df	text	488,944	51,719
	nr_text	486,001	50,879
	l_nr_text	486,005	41,023
	sr_nr_text	308,212	50,214
	sr_l_nr_text	268,138	40,632
dev_1C_df	text	35,975	9,495
	nr_text	35,772	9,367
	l_nr_text	35,772	6,924
	sr_nr_text	22,537	8,851
	sr_l_nr_text	19,657	6,607
test_1C_df	text	140,677	23,301
	nr_text	139,823	22,972
	l_nr_text	139,825	17,484
	sr_nr_text	89,168	22,356
	sr_l_nr_text	77,634	17,124

Table 8: Total and unique token counts for Task 1C at each preprocessing stage.

6 Conclusion

This work explored various machine learning models to detect hate speech in Bengali. The experiments demonstrate that LR and SVM perform well for the downstream task. Hierarchical classification combined with TF-IDF vectorization improves minority-class detection, while majority voting ensembles enhance robustness and overall F1-micro, achieving **68.42%**. Error analysis shows that subtle, context-dependent, and overlapping classes remain challenging, highlighting the value of hierarchical and ensemble strategies in handling nuanced hate speech. In the future, we plan to investigate deep learning and transformer architectures to more effectively identify contextual, implicit, and nuanced hate expressions that traditional methods often miss. We will also leverage sarcasm-aware, context-sensitive models to enhance the detection of subtle and overlapping hate categories.

Limitations

This section summarizes the main limitations of our study and the steps taken to address them:

- **Class Imbalance:** Rare classes like *Sexism* and *Severe* are under-represented, affecting generalization.

- **Contextual and Sarcastic Language:** Subtle or sarcastic expressions are often misclassified due to reliance on lexical features.
- **Domain Limitation:** Models trained only on YouTube comments may not generalize to other Bangla social media.
- **Sparse Vector Dependence:** Models trained on sparse representations (such as TF-IDF) struggle with complex linguistic variations and long-range dependencies.

Acknowledgments

This work is supported by the NLP Lab, Chittagong University of Engineering & Technology (CUET), Chittagong, Bangladesh. We also thank the organizers of BLP 2025 for conducting this shared task.

References

Areej Al-Hassan and Hmood Al-Dossari. 2022. Detection of hate speech in arabic tweets using deep learning. *Multimedia systems*, 28(6):1963–1974.

Md Alam, Hasan Mesbaul Ali Taher, Jawad Hossain, Shawly Ahsan, and Mohammed Moshiul Hoque. 2024. CUET_NLP_Manning@LT-EDI 2024: Transformer-based approach on caste and migration hate speech detection. In *Proceedings of the Fourth Workshop on Language Technology for Equality, Diversity, Inclusion*, pages 238–243, St. Julian's, Malta. Association for Computational Linguistics.

Safae Sossi Alaoui, Yousef Farhaoui, and Brahim Aksasse. 2022. Hate speech detection using text mining and machine learning. *International Journal of Decision Support System Technology (IJDSST)*, 14(1):1–20.

Girma Bade, Olga Kolesnikova, Grigori Sidorov, and José Oropeza. 2024. Social media hate and offensive speech detection using machine learning method. In *Proceedings of the Fourth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 240–244.

Muhammad Bilal, Atif Khan, Salman Jan, Shahrulniza Musa, and Shaukat Ali. 2023. Roman urdu hate speech detection using transformer-based model for cyber security applications. *Sensors*, 23(8):3909.

Patricia Chiril, Farah Benamara Zitoune, Véronique Moriceau, and Abhishek Kumar. 2019. The binary trio at SemEval-2019 task 5: Multitarget hate speech detection in tweets. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 489–493, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in Bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Udoy Das, Karnis Fatema, Md Ayon Mia, Mahshar Yahan, Md Sajidul Mowla, Md Fayez Ullah, Arpita Sarker, and Hasan Murad. 2023. EmptyMind at BLP-2023 task 1: A transformer-based hierarchical-BERT model for Bangla violence-inciting text detection. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 174–178, Singapore. Association for Computational Linguistics.

Omar Faruqe, Mubassir Jahan, Md Faisal, Md Shahidul Islam, and Riasat Khan. 2023. Bangla hate speech detection system using transformer-based nlp and deep learning techniques. In *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–6. IEEE.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Md Sakib Ul Rahman Sourove, Deeparghya Dutta Barua, Md Fahim, and Md Farhad Alam Bhuiyan. 2025. BanTH: A multi-label hate speech detection dataset for transliterated Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7217–7236, Albuquerque, New Mexico. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Jawad Hossain, Hasan Mesbaul Ali Taher, Avishek Das, and Mohammed Moshiul Hoque. 2023. NLP_CUET at BLP-2023 task 1: Fine-grained categorization of violence inciting text using transformer-based approach. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 241–246, Singapore. Association for Computational Linguistics.

Muhammad Okky Ibrohim and Indra Budi. 2019. Multi-label hate speech and abusive language detection in Indonesian Twitter. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 46–57, Florence, Italy. Association for Computational Linguistics.

Shakir Khan, Mohd Fazil, Vineet Kumar Sejwal, Mohammed Ali Alshara, Reemiah Muneer Alotaibi, Ashraf Kamal, and Abdul Rauf Baig. 2022a. Bichat: Bilstm with deep cnn and hierarchical attention for hate speech detection. *Journal of King Saud University-Computer and Information Sciences*, 34(7):4335–4344.

Shakir Khan, Ashraf Kamal, Mohd Fazil, Mohammed Ali Alshara, Vineet Kumar Sejwal, Reemiah Muneer Alotaibi, Abdul Rauf Baig, and Salihah Alqahtani. 2022b. Hcovbi-caps: Hate speech detection using convolutional and bi-directional gated recurrent unit with capsule network. *IEEE Access*, 10:7881–7894.

Md. Emnul Momin and Hasan Sarker. 2025. Enhancing hate speech detection in bengali language using machine learning and deep learning algorithms. In *2025 2nd International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*, pages 1–6, Los Alamitos, CA, USA. IEEE Computer Society.

Sarah Nasir, Ayesha Seerat, and Muhammad Wasim. 2024. Hate speech detection in roman urdu using machine learning techniques. In *2024 5th International Conference on Advancements in Computational Sciences (ICACS)*, pages 1–7. IEEE.

Sagor Kumar Saha, Afrina Akter Mim, Sanzida Akter, Md. Mehraz Hosen, Arman Habib Shihab, and Md Humaion Kabir Mehedi. 2024. Bengalihatecb: A hybrid deep learning model to identify bengali hate speech detection from online platform. In *2024 6th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pages 439–444. IEEE.

Hind Saleh, Areej Alhothali, and Kawthar Moria. 2023. Detection of hate speech using bert and hate speech word embedding with deep model. *Applied Artificial Intelligence*, 37(1):2166719.

Mahamat Saleh Adoum Sanoussi, Chen Xiaohua, George K Agordzo, Mahamed Lamine Guindo, Abdullah MMA Al Omari, and Boukhari Mahamat Issa. 2022. Detection of hate speech texts using machine learning algorithm. In *2022 IEEE 12th annual computing and communication workshop and conference (CCWC)*, pages 0266–0273. IEEE.

A Error Analysis

To assess the model's efficacy in various tasks, a detailed error analysis is conducted.

- **Quantitative Analysis:** Confusion matrices are used to conduct the quantitative analysis that highlights misclassification patterns. For **hate type**, the majority class *None* was well predicted (4827), but minority classes like *Sexism* were often confused with *None* or *Abusive* (Figure 3).

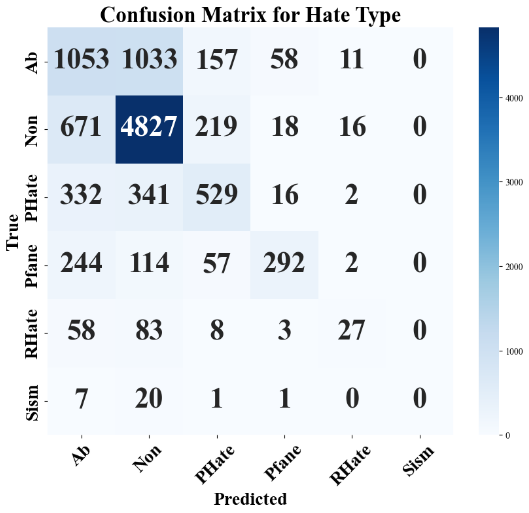


Figure 3: Majority vote ensemble Confusion matrix for Hate Type.

Hate severity showed reliable detection for *Little to None* (6325), while *Mild* and *Severe* were underestimated (Figure 4).

Target group predictions were strong for *None* (5293), but overlapping categories (e.g., *Community* vs. *Individual*) were frequently misclassified (Figure 5).

These patterns result from several challenges. Extreme label imbalance leads models to favor the majority classes, making small but key categories such as *Sexism*, *Severe*, and *Community* more likely to be misclassified. Many comments express hate implicitly, indi-

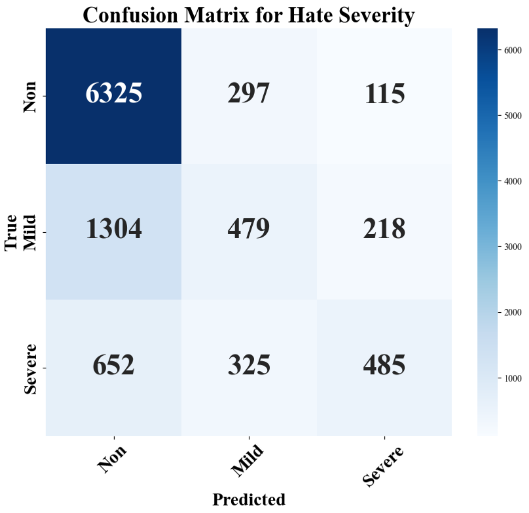


Figure 4: Majority vote ensemble Confusion matrix for Hate Severity

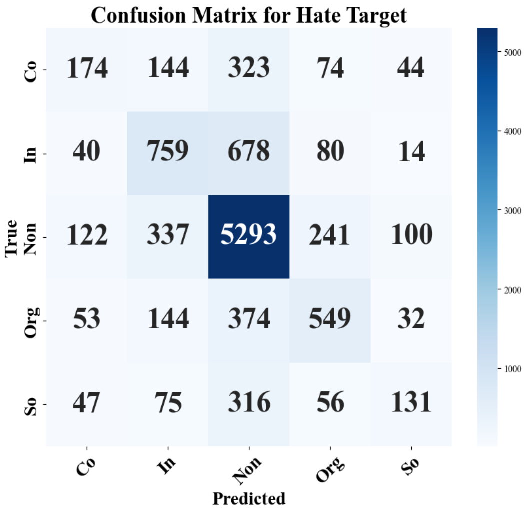
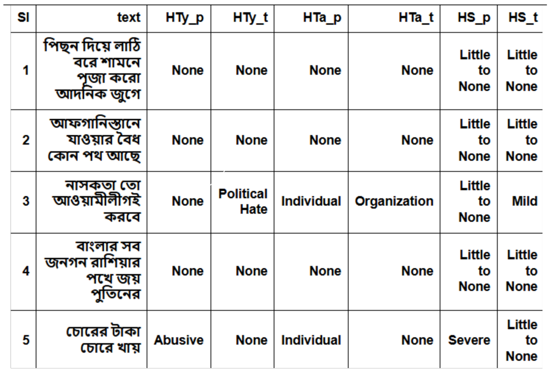


Figure 5: Majority vote ensemble Confusion matrix for Hate Target

rectly, or through sarcasm, which sparse TF-IDF features fail to capture, leading to confusion between labels such as *Mild*, *Little*, and *None*. Target group boundaries often blur in real user comments, leading to ambiguity and errors, such as predicting *Community* as *Individual*. These insights show the limits of classical models in handling nuanced, context-dependent hate speech and reinforce the value of the hierarchical and ensemble strategies used in this study.

- **Qualitative Analysis:** Examining predictions across **HTy, HTa, HS** (Figure 6) reveals systematic patterns.



Sl	text	HTy_p	HTy_t	HTa_p	HTa_t	HS_p	HS_t
1	পিছন দিয়ে লাঠি বরে শামনে পুজা করো আদনিক জুগে	None	None	None	None	Little to None	Little to None
2	আফগানিস্তানে যাওয়ার বৈধ কোন পথ আছে	None	None	None	None	Little to None	Little to None
3	নাসকতা তো আওয়ামীলীগই করবে	None	Political Hate	Individual	Organization	Little to None	Mild
4	বাংলার সব জনগন রাশিয়ার পখে জয় পুতিনের	None	None	None	None	Little to None	Little to None
5	চোরের টাকা চোরে খায়	Abusive	None	Individual	None	Severe	Little to None

Figure 6: Sample predictions from the Majority Vote Ensemble. HTy = Hate Type, HTa = Hate Target, HS = Hate Severity; suffix p = predicted, suffix t = true label, SI = Sample Index.

Political and abusive content was occasionally misclassified (e.g., SI 3: *Political Hate* predicted as *None*). Implicit targets caused errors (SI 3: *Organization* predicted as *Individual*). Severity distinctions were often missed (SI 3: *Mild* predicted as *Little to None*, SI 5: *Little to None* predicted as *Severe*). Neutral examples were accurately classified (SI 1, 2, 4), but subtle, context-dependent hate remained challenging. These qualitative insights complement quantitative findings, emphasizing the utility of hierarchical and ensemble methods for nuanced class detection.

Velora at BLP-2025 Task 1: Multi-Method Evaluation for Hate Speech Classification in Bangla Text

Sad Yeamin Sayem
BRAC University
Department of CSE
sad.yeamin.sayem@g.bracu.ac.bd

Sabira Rahman
BRAC University
Department of CSE
sabira.rahman1@g.bracu.ac.bd

Abstract

Hate speech detection in Bangla is challenging due to complex morphology, frequent code-mixing, and severe class imbalance across categories such as abuse, sexism, religious and political hate, profanity, and neutrality. The BLP Workshop 2025 Subtask 1A addressed this by classifying Bangla YouTube comments into these categories to support online moderation in low-resource settings. We developed a BanglaBERT-based system with balanced data augmentation and advanced regularization techniques, combined with optimized training strategies for better generalization. On the blind test set, our system achieved a micro F1 score of 0.7013, ranking 21st on the leaderboard. These results indicate that augmentation, robust loss functions, and model refinements can enhance Bangla hate speech detection, though implicit and context-dependent hate speech remains difficult.

1 Introduction

Hate speech detection is the task of automatically identifying harmful or offensive language directed towards individuals or groups. Hate speech can take many forms, including derogatory remarks based on race, religion, gender, politics, nationality, or the use of profanities and abusive expressions. Such content poses risks by reinforcing social inequalities, deepening divides, and in some cases fueling real-world hostility. With the rapid growth of online communication, automated hate speech detection has become an important area of research to support moderation and promote safer digital spaces. Recent work has explored similar challenges in other languages, such as the development of a Slovak hate-speech detection system using a neural-network-based approach applied to comments on social media (Sokolová et al., 2022). However, beyond explicit slurs, many instances of hate speech are context-dependent or subtle. These include sarcastic remarks, coded expressions such as references to 'those people' or 'your kind', indirect political or religious insinuations, and statements whose hateful intent becomes evident only through a conversational or cultural context (Ocampo et al., 2023).

In the context of Bangla, these challenges are amplified. Bangla is the seventh most spoken language in the world, with over 230 million speakers, and online content in Bangla is expanding rapidly (Momin and Sarker, 2025a). However, effective moderation tools tailored for Bangla are still limited. The language itself is morphologically rich (Faridee and Tyers, 2009), frequently mixes with English in social media, and exhibits severe class imbalance in hateful expressions across different categories (Chanda et al., 2016). Research on Bangla hate speech detection began only recently; earlier work relied on traditional machine learning methods, followed by deep learning approaches such as CNNs and LSTMs (Rawal and Asirvatham, 2025; Kumar et al., 2024). More recently, transformer-based models like mBERT (Nozza et al., 2020), XLM-RoBERTa (Singh et al., 2023), and BanglaBERT (Bhattacharjee et al., 2021) have demonstrated promising results, supported by shared tasks such as HASOC (Mandl et al., 2025) and dedicated hate speech benchmarks.

Building on this foundation, our work contributes in three key ways:

- **Balanced data augmentation:** combining undersampling of frequent categories with noisy oversampling of rare ones.
- **Model refinements:** adding techniques like mean pooling, multi-sample dropout, and a linear classifier head, trained with Class-Balanced Focal Loss and R-Drop regularization.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 531–537
December 23, 2025 ©2025 Association for Computational Linguistics

- **Advanced training strategies:** including layer-wise learning rate decay, cosine warmup, gradient checkpointing, mixed-precision training, gradient accumulation, and early stopping.

With these improvements, we achieved a micro-F1 score of **0.7013** on BLP 2025 Subtask 1A. This suggests that combining balanced data augmentation with targeted model refinements can yield measurable improvements in hate speech detection in Bangla, although context-dependent and subtle cases remain difficult. Our code and experimental setup are publicly available to facilitate further research in Bangla hate speech detection[1].

2 Related Work

Hate speech detection has been extensively studied in high-resource languages such as English, Hindi, Urdu, Arabic, etc.(Usman et al., 2025; Jahnavi and Chaturvedi, 2025; Ahmad et al., 2024) Early approaches relied on handcrafted features and traditional classifiers such as Naive Bayes, SVM, and Random Forests(Mullah and Zainon, 2021). While these methods demonstrated the feasibility of automated detection, their inability to capture deeper semantics limited generalization (Pruengkarn et al., 2025). The advent of pre-trained language models, such as BERT and its multilingual variants (Nozza et al., 2020), introduced contextual embeddings that became the dominant approach, offering substantial improvements in robustness and cross-lingual transfer (Papel et al., 2024).

In the Bangla context, systematic research is comparatively recent. Initial efforts focused on word embeddings like Word2Vec, FastText (Arif et al., 2024), and GloVe (Mahmud et al., 2022) in combination with classifiers, which achieved moderate success but struggled with challenges unique to Bangla, including morphological richness, frequent code-mixing with English, and severe class imbalance (Momin and Sarker, 2025b).

Several benchmark data sets have been released to support progress in detecting hate speech in Bangla. BanglaBERT (Bhattacharjee et al., 2021) introduced a dataset covering abusive and offensive content, while the Hate Speech and Offensive Content (HASOC) shared tasks (Mandl et al., 2025) included Bangla as one of the languages, providing labeled data for multilingual evaluation. More recently, BIDWESH (Fayaz et al., 2025) expanded its coverage to dialectal and informal Bangla, broadening the scope of evaluation. The Bangla Language Processing (BLP) Workshop 2025 (Hasan et al., 2025b) provided one of the largest curated hate speech benchmarks to date, standardizing evaluation and encouraging cross-system comparisons. Despite these advances, challenges such as low-resource categories (e.g., sexism, religious hate) and context-dependent expressions remain largely unresolved, motivating further exploration of balanced augmentation and model refinements.

3 Task & DataSet Overview

3.1 Task Overview

The Bangla Multi-task Hate Speech Identification shared task (Hasan et al., 2025a), part of the BLP Workshop 2025, aimed to classify Bangla YouTube comments into six categories as abuse, sexism, religious hate, political hate, profanity, and neutral, to establish a benchmark for developing robust hate speech detection systems in Bangla, where challenges such as code-mixing, complex morphology, and severe class imbalance make moderation particularly difficult. Subtask 1A evaluated the systems on this single-label classification task under realistic low-resource conditions.

3.2 Dataset Description

We constructed the training data set by merging the dataset provided for Subtask 1A of the BLP Workshop 2025 (Hasan et al., 2025b) with another publicly available dataset (Karim et al., 2020) which we selected because its categories align well with the BLP Workshop dataset, allowing for a seamless merge and a more comprehensive training set. To ensure consistency, label categories were normalized using a mapping strategy, where synonymous or overlapping labels (e.g., Political and Geopolitical) were merged under a unified class (Political Hate), and ambiguous labels were refined (e.g., Personal → Abusive, Gender abusive → Sexism). After cleaning and harmonization, the final dataset contains six classes: None (neutral/non-hateful), Abusive, Political Hate, Profane, Religious Hate, and Sexism. The class distribution is highly imbalanced, with 19,955 neutral samples. This imbalance reflects the real-world prevalence of hate categories in Bangla social media and motivates the use of augmentation and loss re-weighting strate-

[1] https://github.com/programophile/Bangla_NLP_workshop_Subtask1A_Velora

gies during training.

The merged training set consists of **38,941 samples** (35,523 from shared & 3,418 from the public dataset). Meanwhile, the blind development test set (`dev_test`) contains **2,512** unlabeled samples reserved for evaluation. Table 2 presents the final class distribution of the merged training set.

4 Methodology

4.1 Models & Workflow

All five models were based on BanglaBERT, with variations introduced in architecture, loss functions, and data augmentation. The main challenge across them was severe class imbalance, with categories such as `Sexism` and `Religious Hate` remaining underrepresented and difficult to classify, consistently limiting performance.

Figure 1 outlines the entire pipeline: the data set is balanced and augmented, tokenized, and encoded with BanglaBERT to produce contextual embeddings. These embeddings undergo mean pooling and multi-sample dropout before classification through a linear head trained with R-Drop and class-balanced focal loss to improve robustness.

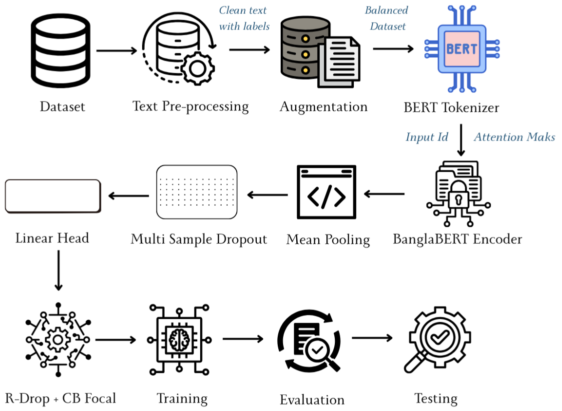


Figure 1: Workflow of the proposed Bangla hate speech classifier: Raw Text → Preprocessing (cleaning & normalization) → Data Augmentation (mitigating class imbalance) → BanglaBERT Encoding (contextual embeddings) → Mean Pooling & Multi-Sample Dropout (generalization) → Classification Head (R-Drop + Class-Balanced Focal Loss), producing robust and fair predictions across all hate speech categories.

4.1.1 Advanced BanglaBERT Fine-tune with CB-Focal & R-Drop

Our submitted approach (Advanced BanglaBERT Fine-tune with CB-Focal & R-Drop) used BanglaBERT as the backbone, leveraging its transformer-based contextual embeddings shown in previous studies to effectively generalize across various Bangla NLP tasks as a strong foundation for classification (Kowsher et al., 2022). The input data was normalized through targeted text cleaning to reduce noise from informal YouTube comments. To address class imbalance, we capped the dominant "None" class via undersampling and duplicated minority class samples with light noisy augmentations. This balanced training exposure prevented the model from collapsing into majority predictions. At the architecture level, we applied mean pooling across token embeddings for efficient sentence-level representation, followed by a single linear classifier head. Generalization was encouraged through multi-sample dropout, which averages predictions across multiple dropout masks, and R-Drop, which enforces consistency across perturbed forward passes. Training optimization combined Class-Balanced Focal Loss, which emphasized difficult minority examples, with layer-wise learning rate decay and cosine scheduling for stable fine-tuning. Training used gradient checkpointing, FP16 mixed precision, gradient accumulation, and early stopping to manage memory usage and stabilize the optimization process.

4.1.2 Balanced Augmented BanglaBERT

Balanced Augmented BanglaBERT introduced two main changes while following the same overall setup as Advanced BanglaBERT Fine-tune with CB-Focal & R-Drop. For data balancing, we replaced simple duplication with controlled duplication combined with word shuffling, which increased sample diversity without redundancy. Inside the model, we replaced the single linear head with a lightweight two-layer MLP, improving convergence stability and reducing overfitting. All other training strategies followed the previous system. These modifications yielded a slightly better result.

4.1.3 BanglaBERT-Hybrid (CNN-BiLSTM-Attn)

This model used standard BanglaBERT fine-tuning with cross-entropy loss. The lack of balancing or augmentation caused a strong bias toward the dominant `None` class and low recall in minority categories. While surface-level hate speech was often detected, context-dependent instances were frequently missed. It achieved a micro-F1 of 0.6954, but without class balancing, it overfit the majority class and showed poor recall for minority labels, resulting in many subtle or contextual hate expressions being misclassified.

4.1.4 Conservative BanglaBERT-Hybrid

This variant extended BanglaBERT with a hybrid CNN-BiLSTM-Attention architecture to capture both local n-gram patterns and long-range dependencies. Training with Class-Balanced Focal Loss improved recall for minority classes, but conservative preprocessing and limited augmentation restricted data diversity, causing underfitting in rare categories. The model achieved a micro-F1 of 0.6793, showing modest but consistent gains over the base system. While minority-class recall improved slightly, precision remained stable and overall recall was still constrained.

4.1.5 Optimized BanglaBERT-Hybrid

This model introduced back-translation, a data augmentation method in which Bangla comments were translated into English and then back into Bangla to generate paraphrased variants that preserved meaning while increasing linguistic diversity. It also applied logit-adjusted cross-entropy to mitigate class imbalance, R-Drop for regularization, and snapshot ensembling for stable predictions. These refinements yielded modest gains and better robustness, though performance remained limited by noisy data and the long-tail label distribution.

4.2 Types of Augmentations Performed on the Minority Classes

Across all variants of the model, the augmentation was applied exclusively to minority hate-speech classes to mitigate the severe class imbalance. The **BanglaBERT-Hybrid (CNN-BiLSTM-Attn)** and **Conservative BanglaBERT-Hybrid** models used light oversampling through direct duplication and controlled word-order shuffling to introduce limited lexical variation. The **Balanced Augmented BanglaBERT** and **Optimized BanglaBERT-Hybrid** systems incorporated stronger augmentation, including targeted oversampling, local word shuffling, and back-translation, particularly for underrepresented categories such as religious hate and sexism, to create semantically diverse paraphrases. The **Advanced BanglaBERT Fine-tune submitted with CB-Focal & R-Drop** applied only light noise-based duplication while capping the dominant None class to maintain balance. Overall, augmentation evolved from simple duplication to richer semantic transformations, consistently targeting minority labels.

5 Results & Findings

The task was evaluated using micro-F1, which balances performance across all classes. As shown in Table 1, our primary model **Advanced BanglaBERT with CB-Focal & R-Drop** achieved 0.7013. The best-performing system, **Balanced Augmented BanglaBERT**, improved this to 0.7025 through controlled augmentation and a lightweight classifier, showing that careful balancing strategies provide more benefit than architectural complexity.

Hybrid models such as **BanglaBERT-Hybrid** and the **Optimized Hybrid** underperformed compared to simpler transformer-based setups, while the **Conservative Hybrid** was lowest. Overall, results highlight that balanced augmentation yields modest but consistent improvements, while rare-class scarcity remains the main bottleneck.

Models	**F1-Score**
Balanced Augmented BanglaBERT	0.7025
Advanced BanglaBERT	0.7013
BanglaBERT-Hybrid	0.6954
Optimized BanglaBERT-Hybrid	0.6886
Conservative BanglaBERT-Hybrid	0.6793

Table 1: micro-F1 scores of different systems

5.1 Category-wise Performance

To understand model behavior across individual classes, we present a comprehensive performance table (Table 3) for the best model **Balanced Augmented BanglaBERT**. These metrics are computed on the validation set used for model selection, and serve to illustrate strengths and weaknesses per category. Final evaluation results reported in Table 1 are based on the test set.

5.2 Error Analysis

Although **Advanced BanglaBERT Fine-tune with CB-Focal & R-Drop**, incorporated strong embeddings, tailored loss functions, and advanced optimization, its effectiveness was limited. Oversampling by simple duplication reduced training diversity and encouraged overfitting. Most critically, categories such as sexism and religious hate remained severely underrepresented, causing low recall despite reweighted loss functions. These factors constrained generalization, leaving implicit or context-dependent hate speech difficult to detect.

In **Balanced Augmented BanglaBERT**, improvements over the submitted system were small. The more balanced augmentation and simplified architecture increased diversity modestly, but rare classes still lacked sufficient coverage. As a result, recall for underrepresented categories improved slightly, yet overall performance remained limited by data scarcity and linguistic complexity.

We also examined weaker variants to understand performance limitations. The **BanglaBERT-Hybrid (CNN-BiLSTM-Attn)** with micro-F1 0.6954, employed a complex hybrid architecture combining CNN, BiLSTM, multi-head attention, and a deep classifier on top of BanglaBERT. While intended to capture richer representations, the added complexity caused overfitting and unstable training. Aggressive focal loss penalized majority classes excessively, further reducing generalization. Consequently, this model failed to surpass simpler model despite higher model capacity.

The **Conservative BanglaBERT-Hybrid** with micro-F1 0.6793 was under-engineered. Its minimal data augmentation left minority categories underrepresented. The class weighting was limited in range, insufficient to address imbalance, and the shallow CNN–LSTM layers lacked expressive power for nuanced hate speech. This led to underfitting on rare classes, making it the least performing system.

Optimized BanglaBERT-Hybrid (micro-F1 0.6886) added back-translation augmentation, logit-adjusted loss, and snapshot ensembling. Despite these improvements, performance gains were just adequate. Back-translation often produced semantically similar sentences rather than truly diverse ones, limiting benefits for rare categories such as sexism and religious hate. Logit-adjusted loss and ensembling stabilized training and slightly improved recall, but severe underrepresentation remained, causing low class-wise F1 for rare labels. Implicit, context-dependent, or subtle hate speech continued to be misclassified, reflecting the reliance on shallow augmentations and surface-level lexical cues. These results indicate that incremental improvements from loss adjustment and ensembling are insufficient without richer data and stronger coverage of minority classes.

6 Conclusion

In this work, we presented multiple Bangla hate speech detection systems for the BLP 2025 shared task 1A. Our best model, **Balanced Augmented BanglaBERT**, achieved a micro-F1 of 0.7025, demonstrating that careful augmentation and lightweight architectures can outperform more complex hybrids. Persistent imbalance in rare classes continues to cap performance, highlighting the need for richer datasets and context-aware augmentation in future work.

Limitations

Our experiments were conducted under compute and time constraints, which restricted exhaustive hyperparameter tuning and limited the exploration of more advanced augmentation strategies. The dataset, while harmonized from existing resources, remained highly imbalanced, with rare categories such as sexism and religious hate severely underrepresented. Augmentation methods like duplication and back-translation produced limited diversity, constraining generalization. Furthermore, the informal nature of Bangla YouTube comments introduced noise and inconsistencies in preprocessing, which may have affected model robustness. Finally, all systems were trained and evaluated on the shared task dataset; broader validation across larger, more diverse Bangla corpora would strengthen the generalizability of our findings.

References

Ashraf Ahmad, Mohammad Azzeh, Eman Alnagi, Qasem Abu Al-Haija, Dana Halabi, Abdullah Aref, and Yousef AbuHour. 2024. Hate speech detection in the arabic language: corpus design, construction, and evaluation. *Frontiers in Artificial Intelligence*, Volume 7 - 2024.

Md Ariful Islam Arif, Md Mahbubur Rahman, Md. Golam Rabiul Alam, and M. Akhtaruzzaman. 2024. Analyzing the performance of deep learning models for detecting hate speech on social media platforms. *MIST INTERNATIONAL JOURNAL OF SCIENCE AND TECHNOLOGY*, 12(2):29–42.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, Md Saiful Islam, Anindya Iqbal, M. Sohel Rahman, and 1 others. 2021. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. *Findings of the Association for Computational Linguistics*, arXiv:2101.00204.

Arunavha Chanda, Dipankar Das, and Chandan Mazumdar. 2016. Unraveling the English-Bengali code-mixing phenomenon. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 80–89, Austin, Texas. Association for Computational Linguistics.

Abu Zaher Md Faridee and Francis M. Tyers. 2009. Development of a morphological analyser for Bengali. In *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 43–50, Alacant, Spain.

A. H. Fayaz, M. D. Uddin, R. U. Bhuiyan, Z. Sultana, M. S. Islam, B. Paul, and S. Manzoor. 2025. BIDWESH: A bangla regional based hate speech detection dataset. *arXiv preprint arXiv:2507.16183*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Blp 2023 task 1: Hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Enduri Jahnavi and Animesh Chaturvedi. 2025. Hate speech detection in low-resource languages hindi, gujarati, marathi, and sinhala. In *2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS)*, pages 7–12.

Md. Rezaul Karim, Bharathi Raja Chakravarthi, John P. McCrae, and Michael Cochez. 2020. Classification benchmarks for under-resourced bengali language based on multichannel convolutional-lstm network. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 390–399.

M. Kowsher, Abdullah As Sami, Nusrat Jahan Prottasha, Mohammad Shamsul Arefin, Pranab Kumar Dhar, and Takeshi Koshiba. 2022. Bangla-bert: Transformer-based efficient model for transfer learning and language understanding. *IEEE Access*, 10:91855–91870.

Ashwini Kumar, Santosh Kumar, Kalpdrum Passi, and Aniket Mahanti. 2024. A hybrid deep bilstm-cnn for hate speech detection in multi-social media. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 23(8).

Md. Shihab Mahmud, Md. Touhidul Islam, Afrin Jaman Bonny, Rokeya Khatun Shorna, Jasia Hossain Omi, and Md. Sadekur Rahman. 2022. Deep learning based sentiment analysis from bangla text using glove word embedding along with convolutional neural network. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6.

Thomas Mandl, Koyel Ghosh, Nishat Raihan, Sandip Modha, Shrey Satapara, Tanishka Gaur, Yaashu Dave, Marcos Zampieri, and Sylvia Jaki. 2025. Overview of the hasoc track 2024: Hate-speech identification in english and bengali. In *Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '24, page 1–2, New York, NY, USA. Association for Computing Machinery.

Md. Emnul Momin and Hasan Sarker. 2025a. Enhancing hate speech detection in bengali language using machine learning and deep learning algorithms. In *2025 2nd International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*, pages 1–6.

Md. Emnul Momin and Hasan Sarker. 2025b. Enhancing hate speech detection in bengali language using machine learning and deep learning algorithms. In *2025 2nd International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*, pages 1–6.

Nanlir Sallau Mullah and Wan Mohd Nazmee Wan Zainon. 2021. Advances in machine learning algorithms for hate speech detection in social media: A review. *IEEE Access*, 9:88364–88376.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific bert models.

Nícolás Benjamí n Ocampo, Ekaterina Sviridova, Elena Cabrio, and Serena Villata. 2023. An in-depth analysis of implicit and subtle hate speech messages. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1997–2013. Association for Computational Linguistics.

Md. Habibur Rahman Papel, Udoy Chandra Dey, and Toufiq Hasan Turza. 2024. Bangla hate speech detection by embedding and hybrid machine learning algorithms. *World Journal of Advanced Research and Reviews*. PDF at https://wjarr.com/sites/default/files/WJARR-2024-3399.pdf.

Ratchakoon Pruengkarn, Supakpong Jinarat, and Ekkasit Srisukha. 2025. Detecting sarcasm in tweets: A comparative study of deep learning and traditional approaches. In *2025 6th International Conference on Big Data Analytics and Practices (IBDAP)*, pages 165–170.

Purvi Rawal and Anisha Asirvatham. 2025. Hate speech detection using lstm and machine learning models. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*.

Karanpreet Singh, Vajratiya Vajrobol, and Nitisha Aggarwal. 2023. IIC_Team@multimodal hate speech event detection 2023: Detection of hate speech and targets using xlm-roberta-base. In *Proceedings of the 6th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text*, pages 136–143, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

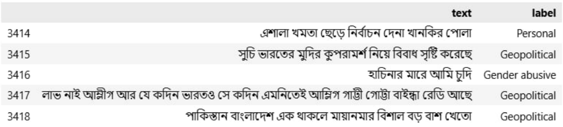
Zuzana Sokolová, Ján Staš, and Daniel Hládek. 2022. An introduction to detection of hate speech and offensive language in slovak. In *2022 12th International Conference on Advanced Computer Information Technologies (ACIT)*, pages 497–501.

M. Usman, M. Ahmad, G. Sidorov, I. Gelbukh, and R. Q. Tellez. 2025. A large language model-based approach for multilingual hate speech detection on social media. *Computers*, 14(7):279.

A Appendix

Category	Count
None	19955
Abusive	8841
Political Hate	6198
Profane	2331
Religious Hate	1178
Sexism	438
Total	38941

Table 2: Class distribution of the merged Bangla hate speech training dataset.



	text	label
3414	এশালা খমতা ছেড়ে নির্বাচন দেনা খানকির পোলা	Personal
3415	সুচি ভারতের মুদির কুপরামর্শ নিয়ে বিবাধ সৃষ্টি করেছে	Geopolitical
3416	হাচিনার মারে আমি চুদি	Gender abusive
3417	লাভ নাই আম্লীগ আর যে কদিন ভারতও সে কদিন এমনিতেই আম্লিগ গাড়ী গোট্টা বাইন্ধা রেডি আছে	Geopolitical
3418	পাকিস্তান বাংলাদেশ এক থাকলে মায়ানমার বিশাল বড় বাশ খেতো	Geopolitical

Figure 2: Sample entries from the Bangla hate-speech dataset introduced by (Karim et al., 2020). The “text” field represents original Bangla social-media posts, and the “label” field provides the manually assigned classification category used for supervised learning tasks.

Category	Precision	Recall	F1-Score
None	0.8356	0.7919	0.8132
Religious Hate	0.2900	0.7632	0.4203
Sexism	0.2000	0.1818	0.1905
Political Hate	0.5895	0.5773	0.5833
Profane	0.7258	0.8599	0.7872
Abusive	0.5809	0.5727	0.5768
Accuracy	–	–	**0.7189**
Macro Avg	0.5370	0.6245	0.5619
Weighted Avg	0.7320	0.7189	0.7232

Table 3: Comprehensive Performance Metrics for Advanced BanglaBERT Fine-tune with CB-Focal & R-Drop model per Category on the Validation Set.

PentaML at BLP-2025 Task 1: Linear Probing of Pre-trained Transformer-based Models for Bangla Hate Speech Detection

Intesar Tahmid[1,*], **Rafid Ahmed**[1,*], **Md Mahir Jawad**[1],
Anam Borhan Uddin[1], **Md Fahim**[1,2,†], **Md Farhad Alam Bhuiyan**[1]

[1]*Penta Global Limited, Bangladesh*
[2]*Center for Computational & Data Sciences*
[*]Equal Contribution [†]Project Lead

Correspondence: {intesar3006, ahmedrafid023, pdcsedu}@gmail.com

Abstract

This paper presents our approach for the BLP Shared Task 1, where we implemented Linear Probing of Pre-trained Transformer-based Models for Bangla Hate Speech Detection. The goal of the task was to customize the existing models so that they're capable of automatically identifying hate speech in Bangla social media text, with a focus on YouTube comments. Our approach relied on fine-tuning several pretrained BERT models, adapting them to the shared task dataset for improved classification accuracy. To further enhance performance, we applied linear probing on three of the fine-tuned models, enabling more effective utilization of the learned representations. The combination of these strategies resulted in a consistent top-15 ranking across all subtasks of the competition. Our findings highlight the effectiveness of linear probing as a lightweight yet impactful technique for enhancing hate speech detection in low-resource languages like Bangla.

1 Introduction

The rapid growth of numerous online platforms has amplified the prevalence of offensive and harmful content, making automatic hate speech detection a pressing challenge in Natural Language Processing (NLP). For low-resource languages like Bangla, the problem is further complicated by the limited availability of large, high-quality annotated datasets.

To advance research in this direction, the organizers of the BLP Shared Task 1 (Hasan et al., 2025a) curated one of the largest annotated corpora for Bangla hate speech detection. Within this framework, transformer-based architectures such as BanglaBERT (Bhattacharjee et al., 2022) have shown considerable promise, owing to their ability to capture nuanced contextual information in Bangla text.

Our work builds upon these advancements by leveraging fine-tuned transformer models alongside linear probing, a lightweight yet effective technique to exploit learned representations for classification. Through systematic experimentation, we demonstrate that this hybrid approach not only improves generalization but also achieves competitive results, securing consistent top-15 rankings across all subtasks of the competition. This paper details our methodology, experimental setup, and key findings, offering insights into the role of linear probing in enhancing hate speech detection in Bangla.

Subtask	Train	Dev	Test	Task & Num of Classes
1A	35,522	2,512	10,200	Single-label (6)
1B				Single-label (5)
1C				Multi-label: (6, 3, 5)

Table 1: Dataset Statistics Across Subtasks

2 Dataset Details

2.1 Task and Dataset Description

The primary objective of this shared task (Hasan et al., 2025a,b) is to advance robust Bangla hate speech detection through multitask learning. This benchmark aims to capture the deeper linguistic and social dimensions of hateful content in Bangla by requiring models to predict not just the presence of hate, but also its type, intensity, and target group.

The shared task consists of three subtasks: Among the three subtasks both 1A and 1B follow a TSV format with columns `id`, `text`, and `label`. The subtask 1C is annotated with three attributes—`hate_type`, `hate_severity`, and `to_whom`. The severity dimension is categorized into *Little to None*, *Mild*, and *Severe*.

Subtask 1A: Given a Bangla text, the model must identify the *type of hate* expressed in it. The possible categories include *Abusive*, *Sexism*, *Religious Hate*, *Political Hate*, *Profane*, and *None*.

Subtask 1B: This subtask focuses on identifying

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 538–543
December 23, 2025 ©2025 Association for Computational Linguistics

the *target group* of hate speech. The label space includes *Individuals*, *Organizations*, *Communities*, and *Society*.

Subtask 1C: The final subtask integrates all previous dimensions in a joint learning setup where hate severity is the new addition to the classification task.
Such a multi-dimensional design encourages the development of systems capable of deeper contextual reasoning rather than keyword-based detection. The data were split into training, validation, and test subsets, maintaining balanced class distributions across all subtasks to support fair and meaningful evaluation.

3 Methodology

3.1 Linear Probing Fine-Tuning

Kumar et al. (Kumar et al., 2022) introduced *Linear Probing Fine-Tuning* (LP-FT), a hybrid training method that combines the benefits of linear probing and full fine-tuning. Their work showed that while full fine-tuning of a model—where both the backbone ϕ_M and classifier head ϕ_C are jointly trained—performs well on in-distribution (ID) tasks, it often fails to generalize to out-of-distribution (OOD) settings. In contrast, linear probing freezes the backbone ϕ_M and trains only the classifier ϕ_C, offering better OOD performance but limited adaptability. LP-FT addresses this by first training only the classifier ϕ_C with the frozen backbone ϕ_M, and then fine-tuning both ϕ_M and the pre-trained ϕ_C jointly on the downstream task. This two-step process improves performance across both ID and OOD scenarios.

3.2 FPT based Linear Probing-Fine Tuning

To adapt the model more effectively to our dataset, we leverage the Linear Probing Fine-Tuning (LP-FT) method (Kumar et al., 2022). Since the pre-trained backbone ϕ_M may not fully capture domain-specific knowledge, we first apply a Further Pre-Training (FPT) step using a Masked Language Modeling (MLM) objective. This step is motivated by insights from BanglaTLit (Fahim et al., 2024) and prior ITPT-based winning strategies (Fahim, 2023).

In MLM, a portion (m%) of the input tokens are randomly masked and replaced with a special token [MASK]. The model is then trained to predict the original tokens using contextual cues from the unmasked tokens. This procedure yields a domain-adapted backbone, denoted as ϕ_M^{FPT}.

Following FPT, we apply LP-FT using ϕ_M^{FPT} and a classification head ϕ_C. We consider two-stage training for FPT-based LP-FT. In the first stage, ϕ_M^{FPT} is frozen and only ϕ_C is trained on the downstream task. Once the classifier has been optimized, in the second stage, we jointly fine-tune both ϕ_M^{FPT} and ϕ_C, allowing the entire model (excluding any frozen parameters) to adapt to task-specific features.

4 Experiment Setup

Experimented Models We experimented with multiple transformer architectures, including BanglaBERT (Bhattacharjee et al., 2022), VACBERT (Bhattacharyya et al., 2023), XLM-RoBERTa (Conneau et al., 2020), and IndicBERT (Dabre et al., 2021). For comparison, we also implemented non-transformer baselines using LSTM and Bi-LSTM architectures with attention mechanisms.

Model Configuration Models were trained for 6 epochs using the AdamW optimizer with betas=(0.9, 0.999), epsilon=1e-6, and weight decay regularization. We set learning rate for the encoder to 2e-5. The batch size was set to 4 for larger transformer models to accommodate memory constraints, while smaller models used a batch size of 16. All inputs were truncated/padded to 256 tokens maximum length.
For all the subtasks, we implemented a cosine learning rate scheduler with warmup steps, gradually decreasing from the initial learning rate to a minimum of 1e-6. For Sub-Task 1C specifically, the training loop utilized BCEWithLogitsLoss for multi-label classification and saved the best-performing model based on validation score. Our code implementation featured differentiated parameter optimization, applying layer-specific learning rates and excluding bias/LayerNorm parameters from weight decay.

Linear Probing Model Configuration For the linear probing setup, we employed an MLM architecture where approximately 13% of the input tokens were randomly masked. The cross-entropy loss between the predicted and true tokens was used as the optimization objective. For all coding and model training processes, the PyTorch framework has been used. All experiments were conducted on Kaggle's P100 GPU infrastructure. We

Models	**Sub-Task 1A**			**Sub-Task 1B**			**Sub-Task 1C**		
	Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
Deep Learning Models									
Non Transformer based									
LSTM	26.19	21.58	18.17	21.02	20.99	18.31	45.68	44.21	44.18
Bi-LSTM	36.97	28.45	28.77	47.21	28.13	29.32	49.43	46.47	46.13
LSTM + Attention	35.83	38.63	36.08	43.07	44.73	43.79	48.79	46.98	47.55
Bi-LSTM + Attention	31.70	33.21	30.10	47.75	45.03	45.57	51.19	47.39	48.44
Transformer based LMs									
IndicBERT	48.46	41.99	64.86	52.24	46.78	65.08	48.39	45.30	46.78
VACBERT	61.07	36.93	62.06	47.53	46.40	63.02	50.26	43.22	45.45
XLM-RoBERTa	54.23	50.64	**70.95**	56.45	57.22	70.09	52.29	50.38	51.68
BanglaBERT	55.41	51.82	70.23	61.70	52.54	71.06	54.29	51.54	52.67
Linear Probing Models									
VACBERT-MLM									
+ Linear Probing	45.34	35.87	61.82	50.13	41.39	61.26	47.38	42.71	44.93
+ Linear Probing-FT	46.58	37.90	63.35	51.10	42.72	63.79	49.43	44.46	46.21
IndicBERT-MLM									
+ Linear Probing	48.56	46.38	64.81	51.35	49.27	65.43	52.36	50.17	50.24
+ Linear Probing-FT	50.54	48.65	66.77	53.60	50.97	67.60	54.29	51.54	52.67
BanglaBERT-MLM									
+ Linear Probing	61.37	53.76	70.58	59.24	53.26	70.03	56.41	51.68	53.62
+ Linear Probing-FT	62.32	55.31	70.88	60.70	54.35	**71.23**	57.59	52.17	**54.22**

Table 2: Model benchmarking results on the **Test split** of the SHARED TASK 1 dataset are reported. **Blue** highlights the highest-performing model and submitted during the competition, while **Cyan** marks the best-performing models for each metric across the model types.

employed a training approach consisting of Further Pre-Training (FPT) using Linear Probing Fine-Tuning (LP-FT).

5 Result and Analysis

Non-Transformer based DL Models. For the non-transformer-based deep learning models, we consider both LSTM and Bi-LSTM architectures. We observe that the Bi-LSTM consistently outperforms the standard LSTM across all subtasks, achieving approximately a 10% improvement in F1 score for Subtask 1A, 11% for Subtask 1B, and 2% for Subtask 1C.

Following the approach proposed by (Yu et al., 2020), we incorporate an attention mechanism on top of both LSTM and Bi-LSTM models for text classification. The addition of attention leads to further performance gains. For the LSTM-based model, F1 scores nearly double for Subtask 1A, and improve by approximately 10% for Subtask 1B and 2% for Subtask 1C. Similarly, the Bi-LSTM model with attention shows improvements of 2% for Subtask 1A, around 15% for Subtask 1B, and 2% for Subtask 1C.

Pretrained LMs Performance We also present a comparative analysis of various transformer-based pretrained language models across all three subtasks. Among the models evaluated, XLM-RoBERTa achieves the highest F1 score for Subtask 1A (70.95), showing a balanced performance in both precision and recall. BanglaBERT also performs competitively, particularly in Subtask 1B and Subtask 1C, where it achieves the highest F1 scores (71.06 and 52.67, respectively), along with strong precision and recall values. VACBERT and IndicBERT show competitive results in precision for Subtask 1A but comparatively lower recall, leading to a moderate F1 score. IndicBERT, while showing consistent performance across subtasks, lags behind in terms of overall F1 scores.

These results demonstrate the effectiveness of multilingual transformer models like XLM-RoBERTa and domain-specific models like BanglaBERT in tackling nuanced classification tasks in Bangla.

Impact of Linear Probing For the linear probing experiments, we first further pretrained the models using MLM, as described in Section 3.2. The models evaluated were VACBERT, IndicBERT, and BanglaBERT. We then applied two training strategies: linear probing alone, and linear probing followed by fine-tuning (FT).

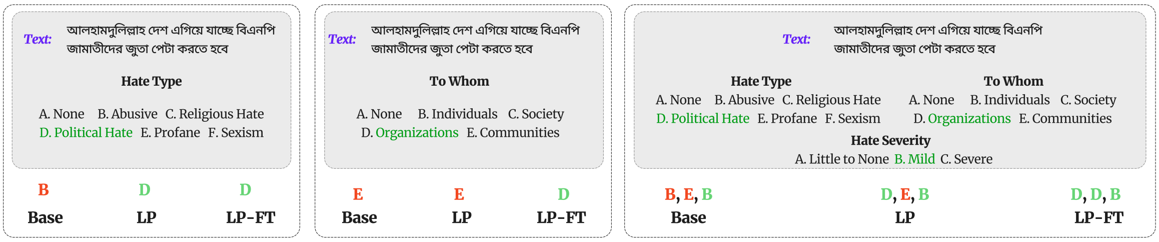

Text:
আলহামদুলিল্লাহ দেশ এগিয়ে যাচ্ছে বিএনপি জামাতীদের জুতা পেটা করতে হবে
Hate Type
A. None B. Abusive C. Religious Hate
D. Political Hate E. Profane F. Sexism
B
Base
D
LP
D
LP-FT
Text:
আলহামদুলিল্লাহ দেশ এগিয়ে যাচ্ছে বিএনপি জামাতীদের জুতা পেটা করতে হবে
To Whom
A. None B. Individuals C. Society
D. Organizations E. Communities
E
Base
E
LP
D
LP-FT
Text:
আলহামদুলিল্লাহ দেশ এগিয়ে যাচ্ছে বিএনপি জামাতীদের জুতা পেটা করতে হবে
Hate Type
A. None B. Abusive C. Religious Hate
D. Political Hate E. Profane F. Sexism
To Whom
A. None B. Individuals C. Society
D. Organizations E. Communities
Hate Severity
A. Little to None B. Mild C. Severe
B, E, B
Base
D, E, B
LP
D, D, B
LP-FT

Figure 1: Error analysis of BanglaBERT variants for hate speech classification, comparing base model, linear probing (LP), and linear probing with full fine tuning (LP-FT) across hate type categories and target demographics.

The results show that linear probing by itself generally underperforms compared to the standard fine-tuning approach presented earlier. However, when linear probing is combined with fine-tuning, the models achieve substantial performance improvements, often surpassing the standard fine-tuning baseline across most models.

Notably, BanglaBERT-MLM with linear probing followed by fine-tuning achieved the highest F1 scores across all subtasks—70.88 for Subtask 1A, 71.23 for Subtask 1B, and 54.22 for Subtask 1C—alongside superior precision and recall in most cases. IndicBERT-MLM also showed notable gains with this combined strategy, consistently outperforming both linear probing alone and standard fine-tuning. Although VACBERT-MLM improved with the combined approach, it still lagged behind the other two models in overall performance.

Therefore, our benchmarking highlights two optimal models tailored to the subtasks: XLM-RoBERTa excels in Subtask 1A, while BanglaBERT-MLM with linear probing followed by fine-tuning (LP-FT) outperforms others in Subtasks 1B and 1C. This suggests that although a powerful multilingual transformer like XLM-RoBERTa is effective for certain tasks, leveraging a monolingual, language-specific model with targeted training strategies such as LP-FT can yield superior results for nuanced tasks in low-resource languages like Bangla. We also provide the result for the validation set in SHARED TASK 1 in Table 3 in the Appendix A.

6 Error Analysis

Figure 1 highlights model confusions across hate type, target, and severity using a representative example. The errors mainly arise from overlapping categories—such as political and abusive hate—and ambiguous phrasing that blurs target boundaries between individuals and communities. The base model often misinterprets these subtleties, while LP improves class awareness through better feature separation. LP-FT further refines contextual understanding, reducing cross-category confusion and capturing implicit hate cues more effectively, thereby aligning predictions closer to human interpretation.

7 Conclusion

In this work, we systematically evaluated Linear Probing followed by Full Fine-Tuning (LP-FT) combined with Further Pre-Training (FPT) for Bengali hate speech classification. Our results demonstrate that BanglaBERT-MLM with LP-FT achieves notable performance gains, highlighting the effectiveness of this two-stage adaptation strategy. This finding establishes that progressive fine-tuning, beginning with a stabilized feature space, is a powerful paradigm for enhancing model robustness in low-resource language contexts, paving the way for more effective and responsible content moderation.

References

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.

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)*, page

1118–1130. Association for Computational Linguistics.

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.

Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh M. Khapra, and Pratyush Kumar. 2021. Indicbart: A pre-trained model for natural language generation of indic languages. *CoRR*, abs/2109.02903.

Md Fahim. 2023. Aambela at blp-2023 task 2: Enhancing banglabert performance for bangla sentiment analysis task with in task pretraining and adversarial weight perturbation. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 317–323.

Md Fahim, Fariha Shifat, Fabiha Haider, Deeparghya Barua, Md Sourove, Md Ishmam, and Md Bhuiyan. 2024. Banglatlit: A benchmark dataset for back-transliteration of romanized bangla. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14656–14672.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Blp 2023 task 1: Hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*.

Shujuan Yu, Danlei Liu, Wenfeng Zhu, Yun Zhang, and Shengmei Zhao. 2020. Attention-based lstm, gru and cnn for short text classification. *Journal of Intelligent & Fuzzy Systems*, 39(1):333–340.

A Result on Validation Set

Table 2 presents a comprehensive comparison of model performances on the validation split of the SHARED TASK 1 dataset, spanning non-transformer deep learning models, transformer-based language models, and linear probing variants.

Among the non-transformer models, the addition of an attention mechanism significantly improves performance. Specifically, Bi-LSTM with attention achieves the highest F1 scores within this group across all subtasks, demonstrating the value of incorporating attention for sequence modeling.

Transformer-based models outperform all non-transformer counterparts, with XLM-RoBERTa achieving the best F1 score in Subtask 1B (57.24), and strong performance across the other subtasks. BanglaBERT shows competitive results as well, particularly excelling in recall metrics, which suggests effective representation learning for Bangla text.

The linear probing experiments further highlight the benefits of task-specific adaptation. While linear probing alone improves over some baseline transformer models, the combination of linear probing with fine-tuning (LP-FT) yields the best results overall. BanglaBERT-MLM with LP-FT achieves the highest F1 scores for Subtasks 1A (56.75) and 1C (55.23), and remains highly competitive in Subtask 1B. IndicBERT-MLM also benefits from this training strategy, consistently improving across metrics.

Highlighted cells denote the best-performing models within each experimental category, as well as the top scores overall. These findings underscore the effectiveness of combining pretrained transformer architectures with targeted fine-tuning approaches, especially for challenging tasks in low-resource languages like Bangla.

Models	**Sub-Task 1A**			**Sub-Task 1B**			**Sub-Task 1C**		
	Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
Deep Learning Models									
Non Transformer based									
LSTM	9.62	16.66	12.20	12.22	20	15.17	13.50	17.80	15.10
Bi-LSTM	41.85	18.13	15.24	36.94	22.15	19.20	38.24	28.50	29.81
LSTM + Attention	35.28	34.12	32.44	40.71	41.04	40.63	45.29	43.93	44.79
Bi-LSTM + Attention	43.62	40.28	42.45	44.19	46.38	47.74	43.51	42.49	43.17
Transformer based LMs									
IndicBERT	46.04	40.86	43.00	51.53	47.16	49.02	47.92	44.31	45.69
VACBERT	43.42	40.46	41.66	46.97	46.38	46.55	47.62	45.07	45.74
XLM-RoBERTa	60.44	54.28	56.58	57.03	57.65	**57.24**	53.74	51.62	52.67
BanglaBERT	56.59	54.90	54.44	59.66	52.81	54.77	54.14	54.60	54.25
Linear Probing Models									
VACBERT-MLM									
+ Linear Probing	47.35	38.73	41.57	51.68	43.54	45.18	47.43	43.89	45.74
+ Linear Probing-FT	48.55	39.42	42.65	52.29	44.20	47.19	49.75	44.79	46.67
IndicBERT-MLM									
+ Linear Probing	53.47	47.39	51.78	52.19	49.63	51.27	53.72	50.16	51.47
+ Linear Probing-FT	54.16	50.98	52.37	53.41	50.83	52.00	55.15	51.00	52.11
BanglaBERT-MLM									
+ Linear Probing	56.78	55.49	55.87	60.17	53.98	56.24	52.87	57.61	54.97
+ Linear Probing-FT	57.50	56.89	**56.75**	60.55	54.27	56.72	53.98	58.21	**55.23**

Table 3: Model benchmarking results on the **Validation split** of the SHARED TASK 1 dataset are reported. **Blue** highlights the highest-performing model and submitted during the competition, while **Cyan** marks the best-performing models for each metric across the model types.

HateNet-BN at BLP-2025 Task 1: A Hierarchical Attention Approach for Bangla Hate Speech Detection

Mohaymen Ul Anam[1*], Akm Moshiur Rahman Mazumder[1*], Ashraful Islam[1], AKM Mahbubur Rahman[1], M. Ashraful Amin[1]

[1]*Center for Computational & Data Sciences, Independent University, Bangladesh (IUB)*

[*]Equal Contribution **Correspondence:** amazumder@iub.edu.bd

Abstract

The rise of social media in Bangladesh has increased abusive and hateful content, which is difficult to detect due to the informal nature of Bangla and limited resources. The BLP 2025 shared task addressed this challenge with subtask 1A (multi-label abuse categories) and subtask 1B (target identification). We propose a parameter-efficient model using a frozen BanglaBERT backbone with hierarchical attention to capture token level importance across hidden layers. Context vectors are aggregated for classification, combining syntactic and semantic features. On subtask 1A, our frozen model with hierarchical attention achieved a micro-F1 of 0.7178, surpassing the baseline of 0.71, while the unfrozen variant scored 0.7149. Our submissions ranked 15th (Subtask 1A) and 12th (Subtask 1B), showing that layer-wise attention with a frozen backbone can effectively detect abusive Bangla text. Our code can be found here https://github.com/MOSHIIUR/BLP_Subtask1A

1 Introduction

Hate speech detection has become an important research problem in Natural Language Processing (NLP) due to its social impact and the increasing spread of harmful content online. Many studies have focused on detecting hate speech, toxic language, and abusive text. While most of these studies have concentrated on high-resource languages like English (Lee et al., 2018; Albladi et al., 2025), low-resource languages such as Bangla have received far less attention (Romim et al., 2022; Das et al., 2022). The Bangla language presents unique challenges due to its complex vocabulary, lack of clear word separation, and regional language variations, making hate speech detection particularly difficult.

The primary objective of this paper is to detect hate speech in Bangla on social media using a layer-wise hierarchical transformer. The Second Bangla Language Processing Workshop (BLP), co-located with IJCNLP-AACL, organized a shared task (Hasan et al., 2025b) and provided a novel dataset for this purpose (Hasan et al., 2025a). The dataset is annotated for multiple subtasks, including subtask 1A, which identifies the type of hate with six classes comprising Abusive, Sexism, Religious Hate, Political Hate, Profane, and None, and subtask 1B, which identifies the target of hate with five classes consisting of Individual, Organization, Community, Society, and None.

To achieve this objective, we have experimented with two transformer-based models: XLM-RoBERTa, and BanglaBERT. Our three main contributions are:

- A parameter-efficient hierarchical attention model built on frozen BanglaBERT, reducing trainable parameters by nearly 89%.
- Improved results on subtask 1A (Micro-F1 = 0.7178), outperforming standard fine-tuning and other baselines.
- An ablation study showing that freezing the backbone improves both performance and efficiency.

2 Related Work

Early studies on Bangla hate speech detection mainly relied on classical machine learning models. For instance, (Alvi and Sharmin, 2019) collected 5,126 Bangla social media comments and achieved 70% accuracy using GRU-based models, outperforming traditional ML approaches. (Romim et al., 2022) created a larger dataset with over 50,000 offensive comments and demonstrated the limitations of standard LSTM and Bi-LSTM models on highly diverse and informal text.

Recently, transformer-based models have shown better performance for Bangla hate speech detection. (Alam et al., 2020) fine-tuned multilingual

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 544–550
December 23, 2025 ©2025 Association for Computational Linguistics

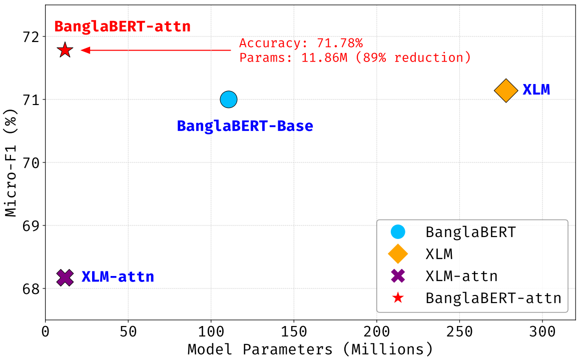


Figure 1: Micro-F1 vs. number of trainable model parameters for subtask 1A on test set. Our proposed BanglaBERT-attn model, using a frozen backbone, achieves the highest accuracy (Micro-F1) while reducing the trainable parameter count by 89% compared to the fully fine-tuned BanglaBERT.

transformers on Bangla text, achieving a 5–29% improvement over previous methods. (Das et al., 2022) developed datasets including both actual and Romanized Bangla posts, where XLM-RoBERTa achieved the best results. Similarly, (Keya et al., 2023) explored BERT-GRU models on Bangla social media comments, showing strong performance.

Recent work has also focused on informal, transliterated, and multi-modal Bangla data. (Islam et al., 2021) collected controversial Bangla social media posts and achieved up to 88% accuracy using SVM. (Karim et al., 2022) explored multimodal hate speech detection with Conv-LSTM and XLM-RoBERTa achieving Micro-F1 scores up to 83%. (Haider et al., 2024) introduced a multi-label Bangla hate speech dataset using translation-based LLM prompting.

3 Methodology

3.1 Dataset Description & Preprocessing

We utilized the dataset offered by the BLP-2025 Shared Task 1 (Hasan et al., 2025a), which consists of manually annotated YouTube comments covering diverse topics such as politics, sports, international news, and major violent incidents. The dataset is structured in a multi-task setup with three subtasks: hate type, hate target, and hate severity. For subtask 1A (hate type), the data include six classes: *Abusive, Sexism, Religious Hate, Political Hate, Profane, or None*. For subtask 1B (hate target), the classes are: *Individual, Organization, Community, Society, or None*. Table 1 summarizes the class-wise distribution across training, development, and test splits for both subtasks.

	Class	Train	Dev	Test
Subtask 1A	Abusive	8212	564	2312
	Sexism	122	11	29
	Religious Hate	676	38	179
	Political Hate	4227	291	1220
	Profane	2331	157	709
	None	19954	1451	5751
	Total	**35422**	**2512**	**10200**
Subtask 1B	Individual	5646	364	1571
	Organization	3846	292	1152
	Community	2635	179	759
	Society	2205	141	625
	None	21190	1536	6093
	Total	**35522**	**2512**	**10200**

Table 1: Overview of the Data and Splitting Procedure for subtask 1A and subtask 1B

For preprocessing, we cleaned the text by removing all non-Bangla characters and punctuation, retaining only Bangla Unicode ranges. This decision was made to reduce input noise and focus the monolingual BanglaBERT model on the lexical features of the Bangla language.

3.2 Method

We propose a hierarchical attention model that leverages multi-level representations learned by pre-trained transformer-based language models. Our model, depicted in Figure 2, introduces a layer-wise attention mechanism to dynamically weigh the importance of hidden representations from each layer of the transformer backbone. The core of our methodology consists of three main stages: (1) a layer-wise attention module, (2) a feature aggregation and projection module, and (3) a final classification head. Hyperparameters for training can be found in Appendix A.1

Problem Formulation: Formally, given a social media comment $S = \{t_1, t_2, ..., t_n\}$ as a sequence of n tokens, the objective is to learn a function $f(S) \rightarrow \hat{y}$. For subtask 1A, $\hat{y}$ is a vector of $K = 6$ binary labels $Y = \{\text{Abusive, Sexism}, \ldots, \text{None}\}$. For subtask 1B (target identification), $\hat{y}$ represents one of $K = 5$ classes $Y = \{\text{Individual}, \ldots, \text{None}\}$. The model is trained to minimize a loss function (e.g., cross-entropy) between the predicted labels $\hat{y}$ and the ground-truth labels y.

Standard fine-tuning of BERT-based models typically addresses this task by feeding the final hidden-state representation of the *[CLS]* token into a classification head. Other common approaches involve pooling the hidden states of all tokens in the final

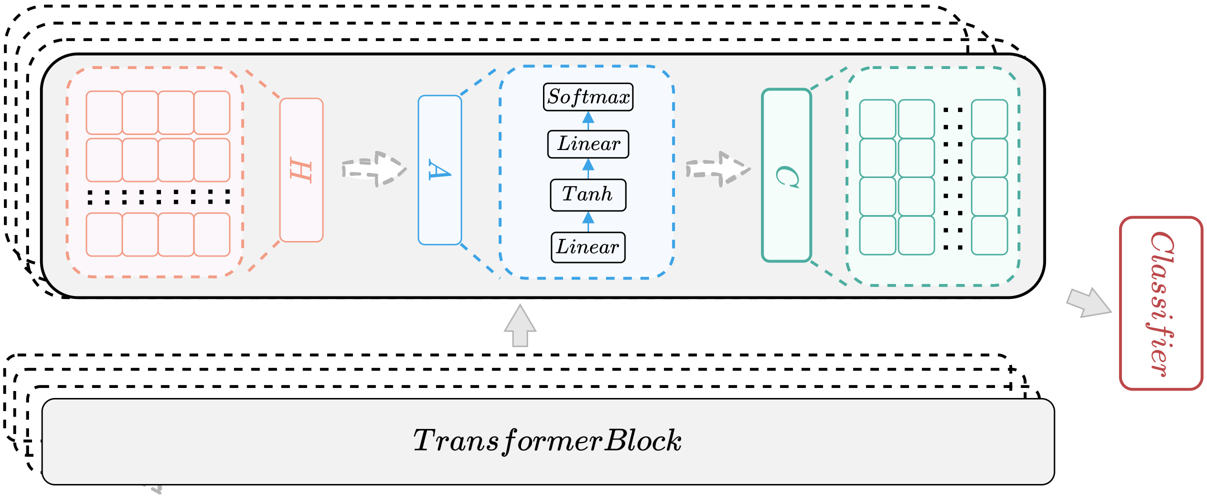


Figure 2: **Architecture of the proposed Hierarchical attention model**. For each of the L layers in the TransformerBlock, the corresponding hidden states, H are processed by a dedicated attention network, A to compute a layer-specific context vector, C. These context vectors are subsequently aggregated and projected into a single feature vector for the final Classifier.

layer. In contrast, our HateNet-BN model hypothesizes that valuable information for hate speech detection is distributed across all layers of the transformer. We posit that intermediate layers capture crucial syntactic and semantic cues that are diluted or lost in the final layer's representation. Our model, therefore, introduces a hierarchical attention mechanism to explicitly weigh and aggregate representations from each layer, creating a richer, multi-level feature vector for classification

Layer-wise Attention Mechanism: Instead of relying solely on the final layer's, our model attends to the representations from all intermediate layers. Let the set of hidden state matrices from the L layers of the transformer be $\{\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \dots, \mathbf{H}^{(L)}\}$, where each $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times d_h}$ contains the hidden states for a sequence of length n, and d_h is the hidden dimension.

We introduce a set of L distinct attention networks, $\{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_L\}$, one for each transformer layer. Each attention network, $\mathcal{A}_l$, is a feed-forward network that learns to assign an importance weight to each token's representation within its corresponding hidden state matrix $\mathbf{H}^{(l)}$.

For a given hidden state $\mathbf{h}_i^{(l)}$ (the i-th token at layer l), the attention network $\mathcal{A}_l$ computes a scalar weight $w_i^{(l)}$:

$$w_i^{(l)} = \mathcal{A}_l(\mathbf{h}_i^{(l)}) \qquad (1)$$

These learned weights are then applied to their corresponding hidden states to produce a weighted representation for that layer. This allows the model to emphasize tokens that are more relevant to the classification task at that specific level of abstraction. A layer-specific context vector, $\mathbf{c}^{(l)} \in \mathbb{R}^{d_h}$, is then generated by computing the weighted sum of the token hidden states:

$$\mathbf{c}^{(l)} = \sum_{i=1}^{n} w_i^{(l)} \mathbf{h}_i^{(l)} \qquad (2)$$

This process is repeated for all L layers, yielding a set of context vectors $\{\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \dots, \mathbf{c}^{(L)}\}$, where each vector encapsulates the most salient information from its respective layer. This layer-wise approach is particularly advantageous for short social media posts. In such condensed texts, meaning is often implicit, and subtle cues (e.g., a single profane word, a sarcastic tone) are critical. By attending to all layers, our model can simultaneously leverage low-level features (like token-level profanity from early layers) and high-level semantic context (like sentence-level sarcasm from deeper layers), which might be overly smoothed in a final-layer-only representation.

Feature Aggregation and Projection: The set of layer-wise context vectors must be aggregated into a single feature vector for the final classification. In this configuration, all layer-wise context vectors are concatenated to form a single, high-dimensional vector, preserving the distinct information from each layer.

$$\mathbf{c}_{\text{concat}} = [\mathbf{c}^{(1)}; \mathbf{c}^{(2)}; \dots; \mathbf{c}^{(L)}] \qquad (3)$$

where $\mathbf{c}_{\text{concat}} \in \mathbb{R}^{L \cdot d_h}$. This vector is subsequently

Subtask 1A

Model	Attn. Net.	Freeze	Abusive	Sexism	Religious Hate	Political Hate	Profane	None	Micro-F1
BanglaBERT	×	×	0.4743	0	0.4225	0.5371	0.6641	0.8246	0.71
BanglaBERT	✓	✓	0.5227	0	0.4677	0.5794	0.6885	0.8283	**0.7178**
BanglaBERT	✓	×	0.4708	0	0.3643	0.5687	0.7202	0.8246	0.7149
XLM-RoBERTa	×	×	0.5172	0	0.4730	0.5743	0.7440	0.8179	0.7114
XLM-RoBERTa	✓	✓	0.4282	0	0.3547	0.5669	0.5938	0.8010	0.6817
XLM-RoBERTa	✓	×	0.5180	0.0606	0.4777	0.5859	0.7478	0.8212	**0.7143**

Subtask 1B

Model	Attn. Net.	Freeze	Individual	Organization	Community	Society	None	Micro-F1
BanglaBERT	×	×	0.6269	0.5757	0.4375	0.4151	0.8299	**0.7187**
BanglaBERT	✓	✓	0.5973	0.5675	0.4068	0.3843	0.8271	0.7148
BanglaBERT	✓	×	0.6315	0.5755	0.4384	0.4173	0.8256	0.7142
XLM-RoBERTa	×	×	0.5976	0.5738	0.4062	0.4201	0.8231	0.7137
XLM-RoBERTa	✓	✓	0.5163	0.5527	0.3341	0.3179	0.8045	0.6857
XLM-RoBERTa	✓	×	0.6084	0.5915	0.4410	0.4204	0.8246	**0.7149**

Table 2: Comparison of Micro-F1 scores between baseline models and our proposed attention network (Attn. Net.) on both subtasks. Results are shown for the attention network with unfrozen(×) and frozen(✓) backbone.

passed through a linear projector layer to reduce its dimensionality back to the model's hidden size, d_h.

$$\mathbf{f} = \mathbf{W}_p \mathbf{c}_{\text{concat}} + \mathbf{b}_p \tag{4}$$

Here, $\mathbf{W}_p \in \mathbb{R}^{d_h \times (L \cdot d_h)}$ and $\mathbf{b}_p \in \mathbb{R}^{d_h}$ are the learnable parameters of the projector.

Classification and Training: The final feature vector $\mathbf{f}$ is fed into a linear classification head, which maps it to the logits for the K target classes.

$$\mathbf{z} = \mathbf{W}_c \mathbf{f} + \mathbf{b}_c \tag{5}$$

where $\mathbf{W}_c \in \mathbb{R}^{K \times d_h}$ and $\mathbf{b}_c \in \mathbb{R}^K$ are the weight and bias parameters of the classifier. The final class probabilities are then obtained by applying the softmax function to the logits:

$$P(y|S) = \text{softmax}(\mathbf{z}) \tag{6}$$

The entire model is trained end-to-end by minimizing the cross-entropy loss $\mathcal{L}$ between the predicted probabilities and the ground-truth labels. For a single training instance, the loss is given by:

$$\mathcal{L} = -\sum_{k=1}^{K} y_k \log(P(y_k|S)) \tag{7}$$

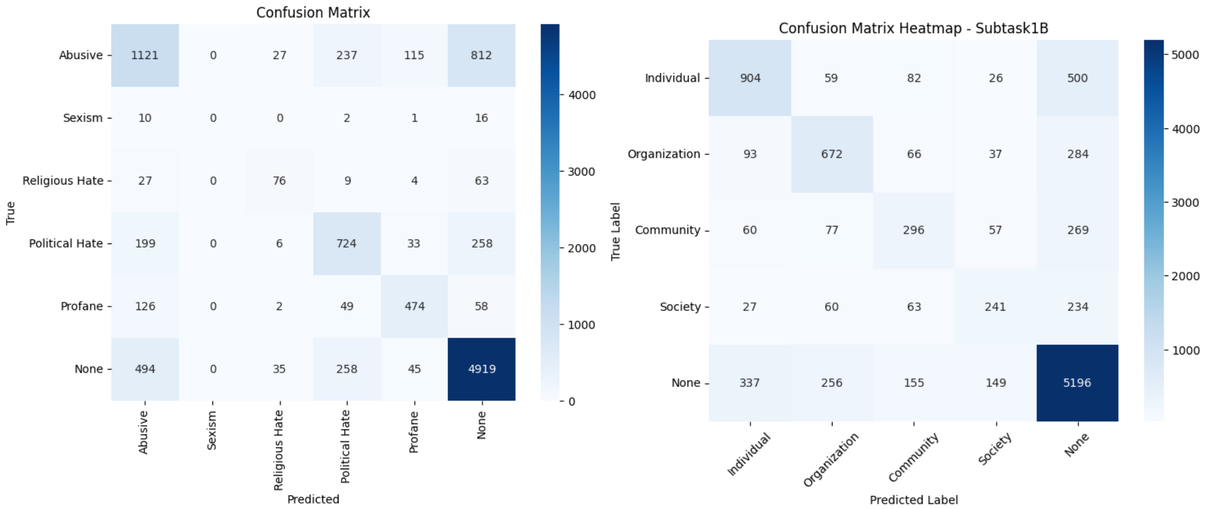
where y_k is a one-hot encoded vector representing the true class label. During training, we can optionally freeze the parameters of the underlying transformer backbone.

4 Results Analysis & Discussion

To evaluate the efficacy of our proposed layer-wise attention mechanism, we conducted a series of experiments on two distinct tasks: subtask 1A and subtask 1B. We benchmarked two transformer-based models, the monolingual BanglaBERT (Bhattacharjee et al., 2022) and the multilingual XLM-RoBERTa (Conneau et al., 2019). For each model, we evaluated three primary configurations: (1) A baseline model using standard fine-tuning without our attention network. (2) Our proposed model with the attention network applied to a fully fine-tuned (unfrozen) backbone. (3) Our proposed model with the attention network applied to a frozen backbone, where only trainable parameters are in our lightweight attention networks ($\mathcal{A}_l$) and the final classification layer (W_c, b_c) which resulting in 89% parameter reduction. For a 12-layer BERT-base model ($L = 12$, $d_h = 768$), this amounts to only 11.86M parameters. The performance of each configuration, measured by Micro-F1 score, is detailed in Table 2.

Performance on Subtask 1A: our proposed attention mechanism demonstrated a clear positive impact, particularly for the monolingual model. BanglaBERT equipped with the attention network and a frozen backbone achieved the highest score of all configurations, with a Micro-F1 of 0.7178.

This represents a notable improvement over its baseline counterpart (0.71). When the backbone was fully trained, the performance of BanglaBERT with our attention network remained strong at 0.7149, still outperforming the baseline. On the other hand, the multilingual XLM-RoBERTa model with standard fine-tuning baseline achieved a competitive score of 0.7114. In contrast to BanglaBERT, adding the attention network while keeping the model frozen led to a significant performance degradation (0.6817). However, with the fully finetuned version it does outperform the



(a) Subtask 1A

(b) Subtask 1B

Figure 3: Confusion matrix for BanglaBERT with hierarchical attention and frozen backbone.

baseline (0.7143).

Performance on Subtask 1B: The results for Subtask 1B reveal a more complex interaction between the models and our proposed method. For BanglaBERT, the standard fine-tuning baseline achieved the highest score of 0.7187. In this case, the introduction of the layer-wise attention network, in both frozen and unfrozen configurations, resulted in a minor decrease in performance. For XLM-RoBERTa, we observe a similar pattern to Subtask 1A. While the baseline result was competitive, adding the attention network to a frozen model again resulted in a comparatively low score of 0.6857. However, the synergy between our attention mechanism and a fully finetuned backbone was once again evident. This variant achieved the highest score among all XLM-RoBERTa configurations for this subtask, reaching a Micro-F1 of 0.7149.

Error Analysis: The confusion matrix for our best-performing model for subtask 1A(Figure 3a) reveals specific strengths and weaknesses. The model excels at identifying the *None* (4919 correct) and *Abusive* (1121 correct) classes. However, significant confusion exists between semantically similar categories. For instance, 126 *Profane* instances were misclassified as *Abusive*, and 199 *Political Hate* instances were also misclassified as *Abusive*. Furthermore, 812 *Abusive* instances were incorrectly labeled *None*, highlighting the difficulty in distinguishing low-severity abusive content from neutral text

Similarly, the confusion matrix for subtask 1B(Figure 3b) shows strong performance on the *None* (5196 correct) and *Individual* (904 correct) target classes. The primary sources of confusion are between *Organization* and *None* (284 misclassifications), and between *Community* and *None* (269 misclassifications), suggesting the model struggles to identify targets when the hateful content is not explicit.

5 Conclusion

In this paper, we introduced HateNet-BN, a parameter-efficient, hierarchical attention model designed to navigate the complexities of Bangla hate speech detection. Our experiments, conducted for the BLP-2025 Shared Task 1, demonstrated that a lightweight attention mechanism can effectively extract and weigh features from the different layers of a large, pre-trained transformer model. Our most significant finding was that the proposed model, when paired with a frozen BanglaBERT backbone, achieved a Micro-F1 score of 0.7178 on Subtask 1A. This result not only surpassed the standard, fully fine-tuned BanglaBERT baseline (0.7104 Micro-F1) but did so while reducing the number of trainable parameters by approximately 89%. This work successfully shows that a surprisingly small set of attention networks can effectively leverage the rich, multi-level representations of a large, frozen language model, offering a solution that is simultaneously high-performing and computationally efficient.

Limitations

Despite its effectiveness, this very efficiency raises critical questions about the scalability of our approach. While effective on the 12-layer architecture of BERT-base models, it is uncertain how this method would scale to significantly deeper models with 24, 48, or more layers. The linear increase in attention networks one for each layer could introduce its own optimization challenges and diminish the parameter-efficiency gains observed here. Future work should therefore investigate strategies for selective or shared attention mechanisms that can maintain this efficiency as model architectures continue to grow in scale.

References

Tanvirul Alam, Akib Khan, and Firoj Alam. 2020. Bangla text classification using transformers. *arXiv preprint arXiv:2011.04446*.

Aish Albladi, Minarul Islam, Amit Das, Maryam Bigonah, Zheng Zhang, Fatemeh Jamshidi, Mostafa Rahgouy, Nilanjana Raychawdhary, Daniela Marghitu, and Cheryl Seals. 2025. Hate speech detection using large language models: A comprehensive review. *IEEE Access*.

Md Ishmam Alvi and Sadia Sharmin. 2019. Hateful speech detection in public facebook pages for the bengali language. In *2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 555–560. IEEE.

Abhik Bhattacharjee, Tahmid Hasan, Kazi Mubasshir, Md. Saiful Islam, Wasi Ahmad Uddin, 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 North American Chapter of the Association for Computational Linguistics: NAACL 2022*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in bengali. 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 1: Long Papers)*, pages 286–296, Online only. Association for Computational Linguistics.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Deeparghya Dutta Barua, Md Sakib Ul Rahman Sourove, Md Fahim, and Md Farhad Alam. 2024. Banth: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Tanvirul Islam, Nadim Ahmed, and Subhenur Latif. 2021. An evolutionary approach to comparative analysis of detecting bangla abusive text. *Bulletin of Electrical Engineering and Informatics*, 10(4):2163–2169.

Md Rezaul Karim, Sumon Kanti Dey, Tanhim Islam, Md Shajalal, and Bharathi Raja Chakravarthi. 2022. Multimodal hate speech detection from bengali memes and texts. In *International Conference on Speech and Language Technologies for Low-resource Languages*, pages 293–308. Springer.

A. J. Keya, M. M. Kabir, N. J. Shammey, M. F. Mridha, M. R. Islam, and Y. Watanobe. 2023. G-bert: An efficient method for identifying hate speech in bengali texts on social media. *IEEE Access*, 11:79697–79709.

Ho-Suk Lee, Hong-Rae Lee, Jun-U Park, and Yo-Sub Han. 2018. An abusive text detection system based on enhanced abusive and non-abusive word lists. *Decision Support Systems*, 113:22–31.

Nauros Romim, Mosahed Ahmed, Md Saiful Islam, Arnab Sen Sharma, Hriteshwar Talukder, and Mohammad Ruhul Amin. 2022. Bd-shs: A benchmark dataset for learning to detect online bangla hate speech in different social contexts. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5153–5162, Marseille, France. European Language Resources Association.

A Appendix

A.1 Hyperparameters

We employed two transformer-based models: XLM-RoBERTa base,BanglaBERT. Our approach involved fine-tuning these models on the preprocessed dataset. Each model was trained for three epochs, a duration sufficient for convergence on the dataset and avoid model overfitting and underfitting. In order to enhance the model's performance, a batch size of 16 was utilized to accelerate the training procedure. The selection of a learning rate of 2e-5 was based on the principle that this rate facilitates more efficient learning of parameter estimates by the algorithm. Table 3 presents the hyperparameter used for this task.

Hyperparameter	**Value**
Learning rate	2e-5
Optimizer	Adam
Batch size	16
Number of epochs	3
Warmup ratio	0.1
Weight decay	0.01
LR scheduler	Cosine
Metric for best model	Micro-F1

Table 3: Hyperparameters used for fine-tuning transformer models

Ecstasy at BLP-2025 Task 1: TF-IDF Informed Prompt Engineering with LoRA Fine-tuning for Bangla Hate Speech Detection

Kazi Reyazul Hasan[1], **Mubasshira Musarrat**[1], **Muhammad Abdullah Adnan**[1]

[1]Department of Computer Science & Engineering,
Bangladesh University of Engineering & Technology,
Dhaka, Bangladesh

Correspondence: kazireyazulhasan@gmail.com, mubasshira31@gmail.com, abdullah.adnan@gmail.com

Abstract

We present a hybrid approach for Bangla hate speech detection that combines linguistic analysis with neural fine tuning. Our method first identifies category specific keywords using TF-IDF analysis on 35,522 training samples. These keywords then inform prompt engineering for Llama 3.1 8B model fine tuned with LoRA adapters. We incorporate distinctive Bangla terms directly into classification prompts to guide the model understanding of hate speech patterns. Our system achieved top 5 rankings across all three BLP 2025 Task 1 subtasks including hate type classification, target identification, and multi task prediction. The approach proved particularly effective for culturally specific hate speech patterns unique to Bangla social media discourse.

1 Introduction

Hate speech detection in Bangla social media presents unique challenges due to the language's complex morphology and culturally specific expressions of hate. The BLP 2025 Task 1 (Hasan et al., 2025b) addresses this critical need by providing a comprehensive dataset of YouTube comments labeled across multiple dimensions of hate speech. This shared task includes three subtasks that progressively increase in complexity. Subtask 1A requires categorizing text into six hate types including Abusive, Sexism, Religious Hate, Political Hate, Profane, or None. Subtask 1B focuses on identifying the target of hate as Individuals, Organizations, Communities, or Society. Subtask 1C combines both tasks in a multi task learning setup.

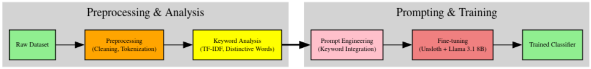


Figure 1: Overview of the Hate Speech Classification Pipeline

We approach these challenges through a unique combination of statistical text analysis and modern language modeling (see Figure 1). Our methodology begins with extensive TF-IDF analysis to identify the most distinctive vocabulary for each hate category. This analysis revealed strong linguistic markers such as religious terms like মুসলিম (muslim), আল্লাহ (allah), and হিন্দু (hindu) for Religious Hate, political party names like লীগ (league) and বিএনপি (BNP) along with ভোট (vote) for Political Hate, and explicit profanity like বাল, শালা for the Profane category. We discovered that certain categories exhibit significantly higher lexical distinctiveness than others. Political and Religious Hate showed average TF-IDF scores above 0.015 for their top keywords, while Abusive and None categories demonstrated more lexical overlap with other classes.

Building on these linguistic insights, we designed category specific prompts that incorporate the identified keywords as examples. This prompt engineering strategy helps the model recognize culturally specific hate patterns that might not be apparent from the text alone. We then fine tuned Llama 3.1 8B using Low Rank Adaptation with rank 64 and alpha 128, training on the full dataset while maintaining computational efficiency through 4 bit quantization. The model processes instructions rather than performing traditional token classification, allowing it to leverage its pretrained knowledge while adapting to Bangla specific hate speech patterns.

Our unified approach achieved competitive performance across all three subtasks, securing top 5 positions in each. The system demonstrated particular strength in identifying explicit profanity with 95 percent recall, though minority classes like Sexism remained challenging due to severe class imbalance. This work contributes both a effective methodology for Bangla hate speech detection and valuable insights into the linguistic patterns of online hate in South Asian social media contexts.

December 23, 2025 ©2025 Association for Computational Linguistics

2 Related Work

Recent advances in Bangla hate speech detection have explored various neural architectures and multilingual models. Faruqe et al. (2023) employed transformer based models including BERT for hate speech classification, achieving high accuracy on social media texts. Mim et al. (2024) investigated ensemble methods combining CNN with traditional machine learning classifiers for multimodal hate detection from videos. There are works that highlighted the challenge of class imbalance in Bangla datasets.

Cross lingual approaches have shown promise for low resource scenarios. Ghosh and Senapati (2025) demonstrated that XLM-RoBERTa fine tuned on Hindi hate speech transfers reasonably to Bangla. Sharma et al. (2025) emphasized the importance of cultural context in South Asian hate speech, showing that generic multilingual models miss region specific slurs and references.

Recent work on prompt engineering for hate detection includes Prome et al. (2025) who used Llama2-7B with carefully crafted prompts for zero shot classification. However, their approach lacked language specific adaptations. Saha et al. (2024) combined lexicon based features with BERT embeddings, achieving improvements on hate detection. Our work differs by systematically extracting category specific keywords through TF-IDF analysis and incorporating them directly into prompts for instruction tuned models, bridging statistical analysis with modern LLM capabilities.

3 Methodology

3.1 Dataset Processing and Class Distribution

The BLP 2025 dataset (Hasan et al., 2025a) consists of YouTube comments exhibiting natural language variations including code mixing, transliteration, and informal spellings common in social media discourse. The training set contains 35,522 samples with severe class imbalance. The None category dominates with 19,954 samples (56.2%), followed by Abusive with 8,212 (23.1%), Political Hate with 4,227 (11.9%), Profane with 2,331 (6.6%), Religious Hate with 676 (1.9%), and Sexism with merely 122 samples (0.3%). This imbalance posed significant challenges for minority class detection. We maintained original distributions during training rather than synthetic balancing to preserve authentic hate speech patterns.

3.2 Keyword Extraction and Analysis

We begin by extracting category specific keywords from the training corpus $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ where x_i represents the text and $y_i \in \mathcal{C}$ denotes the hate category. For each category $c \in \mathcal{C}$, we compute the TF-IDF scores to identify distinctive vocabulary.

The term frequency for word w in document (a full comment here) d is calculated as:

$$\text{TF}(w, d) = \frac{f_{w,d}}{\sum_{w' \in d} f_{w',d}} \tag{1}$$

where $f_{w,d}$ represents the frequency of word w in document d. The inverse document frequency is:

$$\text{IDF}(w, \mathcal{D}) = \log \frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : w \in d\}|} \tag{2}$$

For category specific analysis, we aggregate TF-IDF scores across all documents belonging to category c:

$$\text{Score}(w, c) = \frac{1}{|\mathcal{D}_c|} \sum_{d \in \mathcal{D}_c} \text{TF-IDF}(w, d) \tag{3}$$

We filter keywords appearing in multiple categories using a cross category threshold $\tau = 2$. A word w is retained for category c only if:

$$|\{c' \in \mathcal{C} : w \in \text{Top}_k(c')\}| \leq \tau \tag{4}$$

where $\text{Top}_k(c)$ denotes the top k words for category c. Our analysis identified 316 Bangla stopwords which were removed during preprocessing.

3.3 Prompt Engineering with Keywords

For each hate category c, we construct prompts incorporating the extracted keywords $K_c = \{w_1, w_2, ..., w_m\}$. The prompt template $P(x, K_c)$ is formulated as:

$$P(x, K_c) = \text{Inst} \oplus \bigcup_{c \in \mathcal{C}} \text{Desc}(c, K_c) \oplus x \oplus \text{Label} \tag{5}$$

where Inst represents task instructions, $\text{Desc}(c, K_c)$ provides category description with example keywords, and $\oplus$ denotes concatenation. Each category description includes the top scoring keywords from our TF-IDF analysis.

3.4 Low Rank Adaptation Fine Tuning

We employ LoRA to efficiently fine tune the Llama 3.1 8B model while preserving its general capabilities. The adaptation modifies weight matrices through low rank decomposition:

$$W' = W_0 + \Delta W = W_0 + BA \tag{6}$$

where $W_0 \in R^{d \times k}$ represents frozen pretrained weights, $B \in R^{d \times r}$ and $A \in R^{r \times k}$ are trainable matrices with rank $r \ll \min(d, k)$. We set $r = 64$ and scaling factor $\alpha = 128$.

The training objective minimizes the cross entropy loss over instruction response pairs:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log P(y_i^t | x_i, y_i^{<t}; \theta + \Delta\theta) \tag{7}$$

where $\Delta\theta = \{BA\}$ represents the LoRA parameters. We apply adapters to query, key, value, and output projections in attention layers, as well as the feed forward components.

Training employed gradient accumulation with effective batch size $b_{\text{eff}} = b \times g = 32$ where $b = 8$ is the per device batch size and $g = 4$ is the accumulation steps. We used AdamW optimizer with learning rate $\eta = 2 \times 10^{-4}$ and linear warmup over 10 percent of training steps. The model was quantized to 4 bits using QLoRA for memory efficiency, enabling training on a single Tesla V100 GPU with 32GB VRAM.

3.5 Inference and Prediction

During inference, we generate predictions using greedy decoding with temperature $T = 0$ for deterministic outputs. The predicted category $\hat{y}$ is extracted from the generated text through pattern matching on the instruction following response. For multi task scenarios in Subtask 1C, we parse multiple labels from the structured output format.

4 Results

4.1 Keyword Analysis Findings

Table 1 presents the top distinctive keywords identified through TF-IDF analysis for each hate category on train set. The analysis reveals culturally specific linguistic markers that traditional multilingual models often overlook.

The keyword analysis demonstrates clear lexical separation between categories. Religious and Political Hate exhibit the strongest distinctive vocabularies with average scores exceeding 0.017, while Abusive and None categories show significant overlap with other classes, scoring below 0.008.

Category	Top Keywords (Bangla)	Avg Score
Religious Hate	মুসলিম (0.045), আল্লাহ (0.037), হিন্দু (0.023), ইহুদি (0.022), ইসলাম (0.015)	0.0234
Political Hate	ভোট (0.028), সরকার (0.022), লীগ (0.020), বিএনপি (0.019), আওয়ামী (0.018)	0.0178
Profane	বাল (0.040), শালা (0.015), কুত্তার (0.011), খানকির (0.015), মাদার (0.007)	0.0171
Sexism	নারী (0.042), মহিলা (0.030), মেয়ে (0.021), পুরুষ (0.019), হিজরা (0.017)	0.0221
Abusive	মিথ্যা (0.007), পাগল (0.006), লজ্জা (0.005), চোর (0.008), দালাল (0.013)	0.0080
None	ভাই (0.011), খুব (0.006), দাম (0.005), ঠিক (0.005), সময় (0.007)	0.0069

Table 1: Category-specific keywords with TF-IDF scores

4.2 Classification Performance

able 2 shows the classification results across all three subtasks. Our unified approach achieved competitive performance with consistent results across different hate detection challenges. For Subtask 1C (multi-class classification), we employed a pattern-matching approach where the model directly predicts the next word as the label instead of relying on logits. This method proved more effective, as logits often introduce calibration issues and class imbalance bias, whereas direct next-word prediction aligns better with the generative nature of the model for discrete class outputs.

Subtask	Micro F1	Macro F1	Accuracy
1A: Hate Type	**73.28**	55.6	72.5
1B: Target	**73.17**	55.4	72.3
1C: Multi-task	**73.32**	55.3	72.2

Table 2: Overall performance metrics across subtasks on final test set

4.3 Per-Category Analysis

Detailed classification performance varies significantly across hate categories as shown in Table 3. The model excels at detecting explicit profanity but struggles with minority classes. In the multi-class setting of task 1C, the class imbalances along with multiple output prediction introduce slight confusions between different categories, raising the difficulty for the LLM to disentangle multiple hate indicators within a single utterance but performs better for easier cases.

Category	Precision	Recall	F1	Support
None	81.3	85.2	83.2	1,451
Profane	78.4	**94.9**	85.9	157
Political Hate	58.2	53.3	55.6	291
Abusive	59.1	52.8	55.8	564
Religious Hate	28.6	21.5	24.6	38
Sexism	50.0	18.2	26.7	11
Weighted Avg	**71.8**	**73.2**	**72.3**	2,512

Table 3: Per-category classification performance on validation set (task 1A)

4.4 Ablation Study

We conducted ablation experiments to assess the contribution of each component in our pipeline. Table 4 demonstrates the importance of keyword-informed prompts.

Configuration	Micro F1	∆
Full Model	**73.2**	–
w/o Keyword Prompts	70.4	-2.8
w/o TF-IDF Filtering	71.1	-2.1
w/o Stopword Removal	71.6	-1.6
Base Llama (Zero-shot)	42.3	-30.9
LoRA r=32 (vs r=64)	72.8	-0.4

Table 4: Ablation study showing component contributions on validation set (task 1A)

The ablation results highlight that keyword-informed prompts contribute 2.8 points to the final performance. Removing TF-IDF filtering degrades performance by 2.1 points, indicating the importance of category-specific vocabulary selection. The base model without fine-tuning achieves only 42.3% accuracy, primarily predicting the majority None class.

4.5 Discussion

Our results reveal several insights about Bangla hate speech patterns. The high recall for Profane content (94.9%) suggests that explicit profanity follows consistent linguistic patterns easily captured through keyword matching. Political Hate category benefits substantially from domain-specific vocabulary, explaining their strong TF-IDF scores and reasonable detection rates despite class imbalance.

The poor performance on Sexism and Relgious Hate stems from both data scarcity and subtler linguistic expressions. Unlike explicit profanity, gender-based hate often manifests through context-dependent statements requiring deeper semantic understanding. The model struggles to differentiate between legitimate gender discussions and sexist content, frequently misclassifying them as None.

Error analysis reveals that code-mixed content poses particular challenges. Comments mixing Bangla with English or romanized Bangla often escape detection, as our keyword extraction primarily focused on native script. Additionally, sarcastic or indirect hate speech remains problematic, as the model relies heavily on surface-level keyword indicators rather than contextual interpretation.

The consistent performance across subtasks suggests our approach successfully captures general hate patterns applicable to both type classification and target identification. The performance consistency from Subtask 1A to 1C indicates that multi-task prediction did not introduce additional complexity here.

5 Conclusion

This study presents a comprehensive pipeline for Bangla hate speech classification, integrating linguistic analysis with LLM fine-tuning to address the difficulties of multi-class detection. By identifying category-specific keywords via TF-IDF and incorporating them into structured prompts, our Unsloth-optimized Llama 3.1 8B model achieves a micro F1-score of 72.3% on the validation set. This approach not only enhances model interpretability but also bridges gaps in low-resource language NLP. Future work could extend to real-time deployment and cross-lingual transfer, fostering safer online spaces in Bangla-speaking communities. Our contributions underscore the value of hybrid methods for culturally sensitive moderation.

Limitations

Despite promising results, our model faces challenges from severe class imbalance, leading to confusions with broader content. The reliance on keyword-based prompting may overlook subtle evolving slang, potentially introducing biases from the training corpus. Computational demands of fine-tuning large LLMs limit scalability on resource-constrained devices, and evaluation on a single dataset may not generalize to diverse dialects. Addressing these through balanced augmentation and ensemble methods remains essential for robust, equitable hate speech mitigation.

References

Omar Faruqe, Mubassir Jahan, Md Faisal, Md Shahidul Islam, and Riasat Khan. 2023. Bangla hate speech detection system using transformer-based nlp and deep learning techniques. In *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–6. IEEE.

Koyel Ghosh and Apurbalal Senapati. 2025. Hate speech detection in low-resourced indian languages: An analysis of transformer-based monolingual and multilingual models with cross-lingual experiments. *Natural Language Processing*, 31(2):393–414.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025a. Llm-based multi-task bangla hate speech detection: Type, severity, and target. *arXiv preprint arXiv:2510.01995*.

Md Arid Hasan, Firoj Alam, Md Fahad Hossain, Usman Naseem, and Syed Ishtiaque Ahmed. 2025b. Overview of blp 2025 task 1: Bangla hate speech identification. In *Proceedings of the Second International Workshop on Bangla Language Processing (BLP-2025)*, India. Association for Computational Linguistics.

Shamrin Jahan Mim, Tanjim Mahmud, Md Hasan Ali, and Mohammad Tarek Aziz. 2024. Stacking ensemble framework for hate speech detection in bangla videos. In *2024 IEEE International Conference on Computing, Applications and Systems (COMPAS)*, pages 1–7. IEEE.

Ruhina Tabasshum Prome, Tarikul Islam Tamiti, and Anomadarshi Barua. 2025. Leveraging the potential of prompt engineering for hate speech detection in low-resource languages. *arXiv preprint arXiv:2506.23930*.

Sagor Kumar Saha, Afrina Akter Mim, Sanzida Akter, Md Mehraz Hosen, Arman Habib Shihab, and Md Humaion Kabir Mehedi. 2024. Bengalihatecb: A hybrid deep learning model to identify bengali hate speech detection from online platform. In *2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, pages 439–444. IEEE.

Deepawali Sharma, Tanusree Nath, Vedika Gupta, and Vivek Kumar Singh. 2025. Hate speech detection research in south asian languages: a survey of tasks, datasets and methods. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 24(3):1–44.

CodeAnubad at BLP-2025 Task 2: Efficient Bangla-to-Python Code Generation via Iterative LoRA Fine-Tuning of Gemma-2

Soumyajit Roy
roysoumyajit@icloud.com

Abstract

This paper presents our submission for Task 2 of the Bangla Language Processing (BLP) Workshop, which focuses on generating Python code from Bangla programming prompts in a low-resource setting. We address this challenge by fine-tuning the gemma-2-9b instruction-tuned model using parameter-efficient fine-tuning (PEFT) with QLoRA. We propose an iterative self-improvement strategy that augments the extremely limited training data (74 examples) by reusing verified correct predictions from the development set, alongside LoRA rank experiments (8, 16, 32), observing a clear correlation between rank and accuracy, with rank 32 delivering the best results. Compared to translation-based and retrieval-augmented baselines, our approach achieves significantly higher accuracy, with a pass rate of 47% on the development set and 37% on the hidden test set. These results highlight the effectiveness of combining iterative data augmentation with rank optimisation for specialised, low-resource code generation tasks.

1 Introduction

The ability of large language models (LLMs) to generate code has significantly advanced software development and computer science education. However, most progress has been concentrated on high-resource languages like English. The BLP Task 2 (Raihan et al., 2025c) presents a valuable challenge: extending these capabilities to Bangla, a language spoken by more than 230 million people yet underrepresented in mainstream NLP research. The task requires a system to take a programming prompt written in Bangla and generate a Python script that passes a set of hidden unit tests.

This task is particularly challenging due to two factors: the need for models to comprehend nuanced, procedural instructions in a non-English language, and the extremely limited size of the initial training dataset (74 examples). Our work aims to tackle this by adapting a powerful, pre-trained LLM and introducing a novel training strategy to overcome data scarcity.

Our primary contributions are as follows:

- We introduce an iterative self-improvement pipeline that uses the model's own correct generations on the development set to augment the training data, progressively boosting performance without external data.
- We provide a comparative analysis of different base models (Gemma-2, Code Llama, StarCoder) and LoRA configurations, identifying Gemma-2-9b-it model with a rank of 32 as the most effective combination.
- Our approach achieves a final pass rate of 37% on the official test set and 47% on the development set, showcasing a viable method for low-resource code generation.

2 Related Work

2.1 Large Language Models for Code Generation

Large Language Models (LLMs) have transformed automated code generation. Codex (Chen et al., 2021), powering GitHub Copilot, showcased the potential of training on large-scale code corpora, followed by open-source models like StarCoder (Li et al., 2023) and Code Llama (Touvron et al., 2023). While highly effective in languages such as Python and JavaScript, their performance on low-resource natural languages like Bangla remains underexplored.

Recent efforts target multilingual code generation. Benchmarks such as HumanEval-XL (Peng et al., 2024) and CRUXEval-X (Xu et al., 2025) evaluate cross-lingual generalization and reasoning, while works like Li et al. (2025) and Liu et al. (2025) study zero-shot transfer and retrieval-augmented generation. However, most approaches

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 556–560
December 23, 2025 ©2025 Association for Computational Linguistics

rely on translation or parallel resources, leaving open challenges for direct adaptation in severely low-resource settings. Our work addresses this through iterative fine-tuning on Bangla-specific prompts.

2.2 Parameter-Efficient Fine-Tuning (PEFT)

Fully fine-tuning multi-billion parameter models is computationally prohibitive for most researchers. PEFT methods have emerged as a solution. Low-Rank Adaptation, or LoRA, was introduced by Hu et al. (2021), who proposed freezing the pre-trained model weights and injecting small, trainable low-rank matrices into the transformer layers. This reduces the number of trainable parameters by orders of magnitude. Dettmers et al. (2023) later proposed QLoRA, which applies LoRA on top of a quantized base model, such as 4-bit, making it possible to fine-tune massive models on a single GPU.

2.3 Low-Resource NLP

NLP for low-resource languages like Bangla faces persistent data scarcity, which is even more acute for specialised tasks such as code generation. We address this with an iterative self-improvement approach related to pseudo-labeling, where a model's own predictions are reused as training data. Traditional pseudo-labeling assigns high-confidence predictions to unlabeled data and retrains with thresholds to limit errors (Lee, 2013), while self-training incorporates pseudo-labels without verification (Yarowsky, 1995). In contrast, our method augments training data only with verified predictions from the development set—confirmed via pass rates against hidden unit tests—thereby reducing error propagation in procedural tasks like Bangla-to-Python generation. Unlike general self-training, which risks amplifying noisy labels, our approach establishes a targeted feedback loop that curates high-quality, in-domain pairs without external data, emphasizing execution-based validation over probabilistic confidence for syntactic and semantic accuracy.

3 Dataset and Data Augmentation Strategy

The dataset provided for the task consists of a small training split with 74 samples and a development split with 400 samples (Raihan et al., 2025a), whereas the test dataset consisted of 500 samples (Raihan et al., 2025b). Each sample is a JSON object that contains a Bangla instruction (instruction), a corresponding Python solution (response), and other metadata.

Given the extremely small size of the initial training set, we designed an **iterative self-improvement strategy** to augment our data using the model's own predictions on the development set. This process was executed in the following stages:

- **Initial Training:** The base model was first fine-tuned on the original 74 training samples. This initial model achieved a pass rate of 38% in the 400-sample development set.
- **Data Augmentation:** We identified the 152 correct predictions (38% of 400) from the development set. These high-quality, model-verified instruction-response pairs were then added to the original training data, creating an augmented set of 226 samples.
- **Iterative Re-training:** The model was re-trained from its original checkpoint using this new, larger dataset. This step, combined with training at a higher LoRA rank, improved the development set pass rate from 38% to 42%. A final re-training with an optimised LoRA rank further boosted this score to 47%.

This iterative data curation strategy was central to our ability to improve performance despite the initial data scarcity.

4 Methodology

Our approach is centered around the supervised fine-tuning of a pre-trained LLM using an iterative data augmentation strategy. All training was conducted on a single NVIDIA A6000 GPU, with each run completing in under five minutes due to the small dataset and an early stopping callback monitoring validation loss.

4.1 Base Model Selection

We conducted preliminary experiments with several open-source models to select the best foundation for our task. Using the development set for evaluation, we found that StarCoder (Li et al., 2023) achieved a pass rate of only 13%, while CodeLlama-3.1-8B-it (Touvron et al., 2023) reached 32%. The Gemma-2-9b-it model demonstrated a superior baseline performance, reaching an accuracy of 38%, justifying its selection.

Model Configuration	Dev Set Pass Rate
Translation-based (mBART + Gemma-2)	6%
RAG (Retrieval-Augmented)	11%
StarCoder (Base)	13%
CodeLlama-3.1-8B-it	32%
Gemma-2-9B-it (r=8)	38%
Gemma-2-9B-it (r=16)	42%
Gemma-2-9B-it (r=32)	47%

Table 1: Performance comparison between different models.

We hypothesise that its strong performance stems from a robust instruction-following capability derived from its pre-training, providing a better foundation for interpreting Bangla prompts. We also attempted to use larger 30B+ parameter models, but they overfit rapidly on the small dataset.

In addition to evaluating specialised code generation models, we explored simpler baseline approaches to establish the necessity of our iterative fine-tuning strategy in this low-resource setting. First, we implemented a translation-based method, where Bangla prompts were translated to English using a pre-trained translation model like mBART (Tang et al., 2020) before feeding them into the base gemma-2-9b-it model for code generation. This approach yielded a pass rate of only 6% on the development set, highlighting the challenges of cross-lingual transfer without targeted adaptation. Second, we tested a RAG pipeline using SentenceTransformer ('paraphrase-multilingual-MiniLM-L12-v2') and a FAISS index. Using CodeLlama-7b as the generator, this k=3 few-shot approach achieved only an 11% pass rate, limited by the scarcity of relevant Bangla-aligned retrieval data. These results underscore the inadequacy of off-the-shelf methods for Bangla-specific code generation, justifying our parameter-efficient fine-tuning method with iterative self-improvement, which significantly outperforms these baselines by achieving up to 47% on the development set.

4.2 Fine-Tuning with QLoRA

We applied QLoRA (Dettmers et al., 2023) to enable efficient fine-tuning of the selected base model.

Quantization. The base model was loaded in 4-bit precision using NF4 quantization from bitsandbytes, with computation performed in bfloat16.

LoRA Configuration. Following the LoRA framework (Hu et al., 2021), we experimented with ranks of 8, 16, and 32, observing pass rates of 38%, 42%, and 47% respectively. Our final model used a rank (r) of 32 and a lora_alpha value of 64, applied to all attention and feed-forward layers.

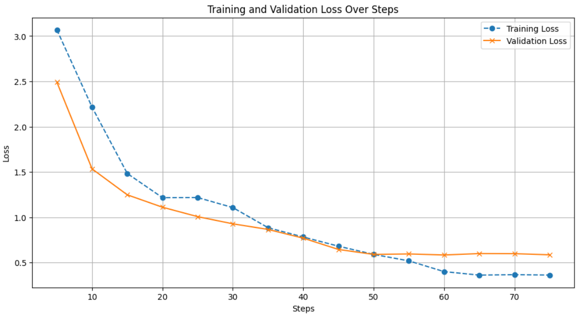


Figure 1: Convergence Graph

4.3 Training

We used the SFTTrainer for finetuning the model. Key parameters included:

- **Optimizer:** Adam (paged_adamw_8bit)
- **Learning Rate:** 5×10^{-5} with a cosine scheduler
- **Effective Batch Size:** 8 (2 per device × 4 gradient accumulation steps)
- **Precision:** bfloat16 with flash attention for faster training

Training was configured with early stopping on validation loss (patience=3), which consistently triggered after a short period, indicating rapid convergence on the small dataset.

5 Results

Our final model achieved a pass rate of 47% on the development set and 37% on the final hidden test set. The iterative data augmentation strategy and experimenting with LoRA rank was critical to our performance, improving the development set pass rate from an initial 38% to 47%—a relative improvement of 24%.

The training and validation loss curve for our final training run in Figure 1 shows that the model began to overfit after approximately 60 steps, and our early stopping mechanism correctly identified the optimal checkpoint, preventing performance degradation.

5.1 Qualitative Analysis

The success of our final model over other configurations can be attributed to two synergistic factors:

- **Superior Base Model Foundation:** The gemma-2-9b-it model's superior performance compared to code-specific models like Code Llama and StarCoder suggests that its robust general instruction-following capabilities, honed during pre-training, provided a better foundation for interpreting the nuanced Bangla prompts. It was more adaptable to the cross-lingual, instructional nature of the task.

- **Targeted Data Strategy:** The iterative self-improvement method was highly effective because it directly addressed the core problem: a lack of training data. Instead of relying on synthetic data from another model, we used our own model's evolving capabilities to curate a high-quality, in-domain dataset. This created a positive feedback loop where each training iteration made the model a better data curator for the next, leading to significant performance gains that would have been impossible with the original 74 samples alone.

5.2 LoRA Rank Experimentation

A key part of our experimentation was finding the optimal LoRA rank (r). We observed a clear correlation between the rank and performance on this dataset:

- A rank of 8 yielded a 38% pass rate.

- A rank of 16 combined with the augmented dataset yielded a 42% pass rate.

- A rank of 32 on this same augmented dataset yielded a 47% pass rate.

Our final and best-performing model used the iterative data augmentation strategy combined with a LoRA rank of 32. This submission achieved a **47% pass rate** on the development set and a **37% pass rate** on the final hidden test set.

5.3 Successful Generations

Analysis of our model's 400 predictions on the development set reveals distinct patterns of success and failure.

The model demonstrated a strong capability for generating complex, multi-part algorithms that require more than just pattern matching. For instance, in problem ID 35, the model was tasked with implementing heap sort. It correctly generated the full algorithm, including a logically sound nested *heapify* helper function.

```
def heap_sort(lst):
    def heapify(arr, n, i):
        # ... (correct heapify logic)

    # ... (correct main sort logic)
    heap_sort_util(lst, len(lst))
    return lst
```

This suggests that the fine-tuning process successfully elicited the base model's underlying algorithmic reasoning capabilities.

5.4 Error Analysis

The 213 failed test cases on the development set can be categorized into several groups. The most common failure mode was the omission of necessary dependencies. For example, in problem ID 26 (calculating a triangle's area), the model generated the mathematically correct formula *(n*n*sqrt(3))/4* but failed with a *NameError* because it did not include the required *from math import sqrt* statement. Similar *NameError* failures occurred for other libraries like *itertools* (ID 91) and re (ID 222). Other frequent errors included *TypeError* (e.g., trying to perform tuple arithmetic), *IndexError*, and logical flaws leading to *AssertionError*. These errors indicate that while the model is adept at generating the core logic, it struggles to consistently produce complete, self-contained, and executable scripts.

6 Conclusion

In this paper, we presented an efficient approach to Bangla-to-Python code generation by fine-tuning Gemma-2-9B with QLoRA and an iterative self-improvement strategy that augments scarce training data using the model's verified outputs. LoRA rank experiments revealed higher ranks improve performance, outperforming translation-based and retrieval-augmented baselines. Our findings highlight iterative fine-tuning and hyperparameter optimisation as practical for low-resource code generation tasks.

For future work, we propose two main directions. First, generating a larger, higher-quality augmented dataset could provide a stronger foundation for training. Second, exploring alternative model architectures, such as dedicated encoder-decoder models, may yield better results for this cross-lingual translation-like task.

Limitations

While our approach demonstrates a viable method for this low-resource task, we acknowledge several key limitations that constrained performance and highlight directions for future work.

Our iterative strategy is fundamentally dependent on the quality and diversity of the small initial dataset (74 samples), which risks amplifying initial data biases. This data scarcity also constrained our model choice to a 9B model, as larger 30B+ parameter models quickly overfit. Furthermore, our study is limited in its baseline comparisons. We did not compare our approach to methods using synthetic data distilled from stronger models, nor did we perform an ablation study to isolate the performance impact of QLoRA quantization against full-precision fine-tuning.

A final limitation, evident from error analysis, is the model's tendency to generate logically correct but syntactically incomplete code, most often causing *NameError* from missing imports (e.g., failing to include *import math*). This highlights a gap between learning core algorithms and the scaffolding (imports, class definitions, etc.) needed for executable scripts.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Lilian Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 38 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Dong-Hyun Lee. 2013. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks.

Mingda Li, Abhijit Mishra, and Utkarsh Mujumdar. 2025. Bridging the language gap: Enhancing multilingual prompt-based code generation in llms via zero-shot cross-lingual transfer. *Preprint*, arXiv:2408.09701.

Raymond Li, Loubna Ben Allal, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Ferrandis, Rohan Ghosh, Niklas Muennighoff, Pratik Mishra, Manan Dey, Rachel Bawden, Ankur Dey, Yacine Jernite, Nouamane Tazi, Julien Launay, Margaret Mitchell, Thomas Wolf, Harm de Vries, Alexander M Rush, and Teven Le Scao. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Wei Liu, Sony Trenous, Leonardo F. R. Ribeiro, Bill Byrne, and Felix Hieber. 2025. Xrag: Cross-lingual retrieval-augmented generation. *Preprint*, arXiv:2505.10089.

Qiwei Peng, Yekun Chai, and Xuhong Li. 2024. Humaneval-xl: A multilingual code generation benchmark for cross-lingual natural language generalization. *Preprint*, arXiv:2402.16694.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *Preprint*, arXiv:2008.00401.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ruiyang Xu, Jialun Cao, Yaojie Lu, Ming Wen, Hongyu Lin, Xianpei Han, Ben He, Shing-Chi Cheung, and Le Sun. 2025. Cruxeval-x: A benchmark for multilingual code reasoning, understanding and execution. *Preprint*, arXiv:2408.13001.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, page 189–196, USA. Association for Computational Linguistics.

Troopers at BLP-2025 Task 2: Reward-Selective Fine-Tuning based Code Generation Approach for Bangla Prompts

Musa Tur Farazi   **Nufayer Jahan Reza**
Department of Computer Science and Engineering (CSE)
Department of Biomedical Engineering (BME)
Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh
{musatur330,nufayerjahanreza8}@gmail.com

Abstract

We present a formally grounded description of a reward-selective fine-tuning (RSFT) pipeline for code generation from Bangla natural-language prompts. The implemented system mines candidate programs via temperature and nucleus sampling, executes candidates in a sandbox and retains programs that pass all unit tests, performs supervised fine-tuning (SFT) on winners using parameter-efficient Low rank adaptation (LoRA) adapters, and augments robustness through fuzzed asserts. We specify the exact objectives and estimators used, provide a Bangla-aware preprocessing recipe, prove simple properties of the sampling budget, and report an ablation showing the effect of inference sample budget K on accuracy. We also include a threat model for safe execution. Our codes are available on GitHub.[1]

1 Introduction

We investigate *reward-selective fine-tuning* (RSFT) of Bangla-to-Python code, a light-weight generate–execute–select–SFT loop that only keeps execution-checked contenders and fine-tunes with maximum likelihood. Unlike RLHF-style policy optimization, RSFT avoids reward modeling and on-policy credit assignment, avoiding instability and engineering overhead. Our system combines stochastic discovery with sandboxed unit tests and fuzzed asserts for safety, and uses LoRA for parameter-efficient adaptation. We then analyze the sample budget K, showing why returns saturate, and see PASS@1/PASS@k ablations as predicted by the theory. Our main contributions in this instance are an industrial RSFT pipeline for Bangla code, fuzzed asserts hardening execution harness, and a slim theory–experiment bridge for the sample budget role.

2 Related Work

Policy-gradient methods and RLHF optimize non-differentiable or preference rewards but are complex and unstable (Ranzato et al., 2016; Paulus et al., 2018; Stiennon et al., 2020; Ouyang et al., 2022).Generate–filter–finetune schemes avoid policy gradients by selecting high-quality samples (RAFT/RSFT; STaR) or by converting preferences into supervised loss (DPO) (Dong et al., 2023; Zelikman et al., 2022; Rafailov et al., 2023). LoRA updates a tiny fraction of weights and pairs naturally with RSFT (Hu et al., 2021). For code models, large-scale supervised/instruction tuning and verification-guided training underpin systems such as Codex, AlphaCode, Code Llama, and CodeRL (Chen et al., 2021; Li et al., 2022; Rozière et al., 2023; Le et al., 2022).We adopt the RSFT recipe with execution correctness as the filter and LoRA for efficient updates.

3 Problem Statement

Let $\mathcal{P}$ be the set of Bangla prompts and $\mathcal{Y}$ the set of syntactically valid Python programs (token sequences). With parameters θ,

$$P_\theta(y \mid p) = \prod_{t=1}^{L} P_\theta\big(y_t \mid p, y_{<t}\big), \qquad (1)$$

where $y = (y_1, \dots, y_L)$.

Each prompt p has a unit-test suite T_p. We use a binary reward

$$r(y; T_p) = \mathbf{1}\{y \text{ passes all tests in } T_p\}, \qquad (2)$$

and optionally a fractional reward $r_{\text{frac}}(y; T_p) \in [0, 1]$.

4 Bangla-aware Preprocessing

Unicode normalization. Prompts are normalized to NFC following the Unicode standard to harmonize visually identical but canonically distinct sequences (The Unicode Consortium, 2024).

[1] https://github.com/Musa-Tur-Farazi/BLP-Code-Genenation-Task.git

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 561–565
December 23, 2025 ©2025 Association for Computational Linguistics

Script and punctuation. We preserve Bangla digits and punctuation; ASCII punctuation present in prompts is retained (no transliteration) to avoid corrupting code-like tokens in the target.

Tokenization. Subword tokenization jointly covers Bangla prompts and Python targets using SentencePiece/BPE (Kudo and Richardson, 2018), a choice consistent with recent Bangla language modeling work(Bhattacharjee et al., 2022; Sennrich et al., 2016). We refrain from code-specific token surgerto avoid introducing off-policy artifacts.

5 Mining by Sampling and Sandboxed Execution

Candidates are sampled stochastically and executed under isolation.

Decoding. Let z_t be logits at step t. Temperature $T > 0$ rescales logits to z_t/T. Nucleus (top-p) sampling restricts sampling to the smallest token set with cumulative probability at least p. The miner distribution is $\pi_{\text{old}}(\cdot \mid p)$.

Discovery statistics. Under π_{old},

$$p_{\text{succ}}(p) = \Pr_{y\sim\pi_{\text{old}}(\cdot|p)} \big[r(y; T_p) = 1\big]. \tag{3}$$

With K independent draws, $\mathbb{E}[\text{winners}] = K\, p_{\text{succ}}(p)$ and

$$\Pr(\text{at least one winner}) = 1 - \big(1 - p_{\text{succ}}(p)\big)^K. \tag{4}$$

Threat model and sandboxing. Execution occurs in a restricted environment with no network, including CPU, memory, time limits, constrained file-system and without modules and system calls. Unit tests run solely within this enclave to evaluate $r(y; T_p)$.

6 RSFT Dataset and Supervised Fine-tuning

From the sampled candidates, winners are retained. Let $\mathcal{S}_p$ be the multiset of winners for p. The induced empirical RSFT distribution is

$$Q(y \mid p) = \frac{\pi_{\text{old}}(y \mid p)\, r(y; T_p)}{Z(p)}, \qquad Z(p) = \mathbb{E}_{y\sim\pi_{\text{old}}(\cdot|p)}\big[r(y; T_p)\big]. \tag{5}$$

Supervised fine-tuning minimizes the negative log-likelihood on $\mathcal{D}_{\text{rsft}} = \{(p, y)\}$:

$$\mathcal{L}_{\text{MLE}}(\theta) = -\sum_{(p,y)\in\mathcal{D}_{\text{rsft}}} \log P_\theta(y \mid p). \tag{6}$$

Training on samples from Q corresponds to minimizing

$$\mathbb{E}_p[\text{KL}(Q(\cdot \mid p) \,\|\, P_\theta(\cdot \mid p))] .$$

7 Parameter-efficient Fine-tuning

We use LoRA adapters to reduce trainable parameters. For weight matrix $W \in \mathbb{R}^{d_o\times d_i}$,

$$W' = W + \Delta W, \qquad \Delta W = BA, \tag{7}$$

with $A \in \mathbb{R}^{r\times d_i}$, $B \in \mathbb{R}^{d_o\times r}$, $r \ll \min(d_i, d_o)$. Only A and B are updated.

8 Robustness using Fuzzed Asserts

To discourage brittle solutions, some test suites are augmented with perturbed inputs and mutated asserts. If $\mathcal{T}'_p$ denotes the augmented set,

$$r'(y; \mathcal{T}'_p) = \mathbf{1}\{y \text{ passes all tests in } \mathcal{T}'_p\},$$

which tightens the correctness predicate and improves selection precision.

9 Evaluation Metrics

We report PASS@1 with greedy decoding and PASS@k with stochastic sampling following the HumanEval/Codex protocol (Chen et al., 2021). Given n samples with c correct,

$$\widehat{\text{pass@}k} = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}, \qquad n \geq k, \tag{8}$$

the standard unbiased estimator under sampling without replacement. For uncertainty we use exact Clopper–Pearson or Wilson score intervals (Clopper and Pearson, 1934; Wilson, 1927).

10 Formal Properties of the Sampling Budget K

Let $p = p_{\text{succ}}(p)$ for brevity. The function $f(K) = 1 - (1-p)^K$ (Eq. 4) is:

- **Monotone** in K for any $p \in (0, 1]$.
- **Concave** in K (diminishing marginal returns), since $f''(K) = -(1-p)^K \ln^2(1-p) \leq 0$.
- To attain $\Pr(\geq 1 \text{ winner}) \geq 1 - \delta$, it suffices that

$$K \;\geq\; \frac{\ln \delta}{\ln(1-p)} \;\approx\; \frac{1}{p} \ln\frac{1}{\delta} \quad \text{for small } p.$$

These properties explain the empirical concavity with the change in K.

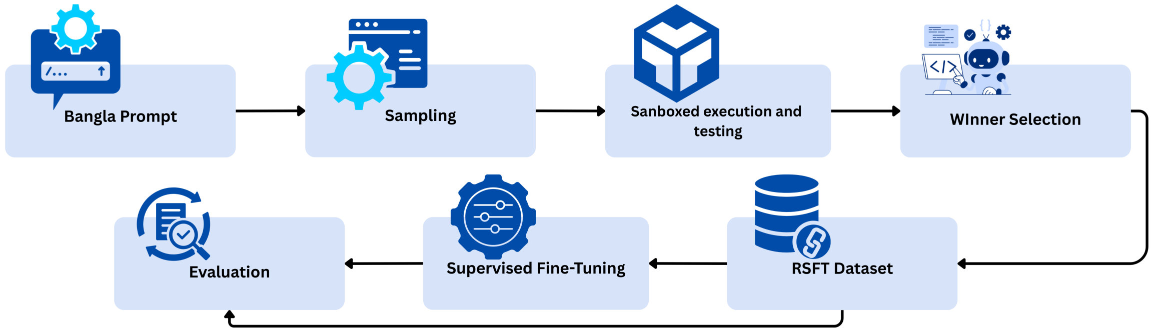


Figure 1: RSFT pipeline: sampling → sandboxed execution → winner selection → SFT with LoRA → evaluation.

11 Pseudocode (RSFT Mining & SFT)

Algorithm 1: RSFT Mining

```
Input: Prompt set $\mathcal{P}$; generator $\pi_{\text{old}}(\cdot \mid p)$;
       sandboxed executor $T_p$; optional
       scorer $s(p, y)$;
Output: Mined supervision pairs $D_{\text{rsft}}$
$D_{\text{rsft}} \leftarrow \varnothing$
foreach $p \in \mathcal{P}$ do
    Sample $y^{(1:K)} \sim \pi_{\text{old}}(\cdot \mid p)$
    $S \leftarrow \{\, y^{(j)} \,:\, \textsc{Pass}(T_p(y^{(j)}))\,\}$
    if $s$ is provided then
        choose $W \subseteq S$ of size $m$
          maximizing $s(p, y)$
    else
        choose $W \subseteq S$ of size $m$
    end
    $D_{\text{rsft}} \leftarrow D_{\text{rsft}} \cup \{(p, y) : y \in W\}$
end
return $D_{\text{rsft}}$
```

Algorithm 2: Fine-tuning

```
Input: Base model $f_\theta$ with LoRA adapters;
       dataset $D_{\text{rsft}}$; optimizer $\mathcal{O}$; batch
       size $B$; epochs $E$
Output: Adapted parameters $\theta^\star$

for $e \leftarrow 1$ to $E$ do
    for mini-batch $\mathcal{B} \subset D_{rsft}$ of size $B$ do
        compute $\mathcal{L}_{\text{MLE}}(\theta; \mathcal{B}) =$
          $-\sum_{(p,y)\in\mathcal{B}} \log p_\theta(y \mid p)$
        backpropagate $\nabla_\theta \mathcal{L}_{\text{MLE}}$; update
          LoRA parameters using $\mathcal{O}$
    end
end
return $\theta^\star$
```

12 Datasets

We follow the BLP-2025 Task 2 split (Raihan et al., 2025c) with an organizer-provided *trial* set for format checks, *mHumanEval-Bangla* for development (Raihan et al., 2025a), and *MBPP-Bangla* for held-out evaluation (Raihan et al., 2025b). All prompts are Bangla (`instruction`); unit tests are Python snippets stored in `test_list` and executed in a sandbox. The test split contains hidden tests available only at scoring time.

Each row contains an `id`, a Bangla prompt `instruction`, optional `response` (trial), and Python tests in `test_list`.

Datasets	Row Count	Columns
Trial	74	id, instruction, response, testlist
Dev	400	id, instruction, testlist
Test	500	id, instruction, testlist

Table 1: Dataset splits and schema.

13 Experimental Findings

We varied the inference sampling budget K and evaluated PASS@1 locally:

Sample Budget K	Passes	Total	PASS@1 (%)
10	176	500	35.20
20	192	500	38.40
50	227	500	45.40
100	245	500	49.00

Table 2: Ablation on inference sampling budget K (higher is better). Results reported as number of tasks passed out of 500 and PASS@1 (%).

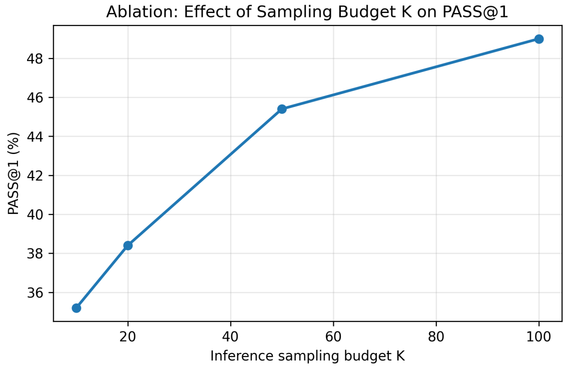


Figure 2: Ablation: accuracy vs. inference sampling budget K.

However, we could achieve a maximum PASS @ 1 score of 31. 6 % (with K = 100) in the online hidden test environment, suggesting that our generated codes were vulnerable to many other edge cases to execute properly.

All experiments ran on an **NVIDIA GeForce RTX 3050** GPU with limited capacity. Unless otherwise stated, we kept mining temperature/top-p, number of mining samples per prompt, LoRA rank and target modules, learning rate, batch size, and epochs constant across the ablation.

Parameter	**Value**
Max Sequence Length	1024
Batch Size (Train/Eval)	16
Gradient Accumulation Steps	4
Max Steps	60
Learning Rate	5×10^{-5}
Weight Decay	0.04
Warmup Steps	10%
Optimizer	AdamW (8-bit)
LR Scheduler	Cosine
Precision	BF16
Seed	3407

Table 3: Hyperparameters used for training, RSFT, and inference.

We observe four recurring classes of failures during development inference for several runs locally:

- **Type-1. Specification misinterpretation:** partial or incorrect adherence to the Bangla prompt (e.g., missing edge conditions, misread constraints).
- **Type-2. I/O contract violations:** mismatch between required and produced interfaces (return vs. print, extra prompts, stray debug output).
- **Type-3. Numerical/algorithmic edge cases:** brittle handling of boundary values (integer vs. float semantics, off-by-one loops, corner-case arithmetic).
- **Type-4. Resource/pathological behavior:** nonterminating or superlinear routines that exceed time/memory limits under hidden tests.

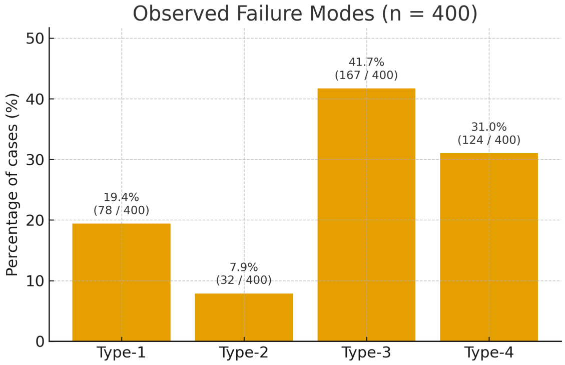


Figure 3: Observed failure modes (Types 1–4) across 400 samples. Labels show percentage and counts.

14 Conclusion

We introduced a compact RSFT pipeline for Bangla-to-Python code generation that integrates sampling, sandboxed execution, winner-only supervision, and LoRA-based adaptation. On this task, we observed diminishing returns with larger sampling budgets and identified recurrent failure modes that motivate tighter verification. The approach is simple, reproducible, and safety-conscious via unit-test gating and fuzzed asserts. Future work includes stronger test generation, adaptive mining of K by prompt difficulty, and richer program-analysis signals to improve generalization.

15 Limitations and Ethics

Model quality is bounded by the coverage and rigor of unit tests hence behaviors outside the test distribution may persist, and mining can overfit to artifacts of a particular decoding configuration (e.g., temperature/top-p), favoring shorter or more verbose programs. The generate–execute–select loop also induces selection bias toward easily verifiable solutions and may under-represent semantically correct but slow or non-deterministic code. Although execution occurs in a hardened sandbox, residual risks remain (e.g., resource exhaustion). We therefore adopt defense-in-depth and strict time/memory limits. Due to our limited compute resources, results are further constrained, which restricts hyperparameter sweeps and ablation breadth and may increase variance.

References

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Kazi Samin, 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. *Preprint*, arXiv:2101.00204.

Mark Chen and 1 others. 2021. Evaluating large language models trained on code. *arXiv:2107.03374*.

Charles J. Clopper and Egon S. Pearson. 1934. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413.

Yao Dong and 1 others. 2023. RAFT: Reward ranked finetuning for generative foundation models. *arXiv:2304.06767*.

Edward J Hu and 1 others. 2021. LoRA: Low-rank adaptation of large language models. *arXiv:2106.09685*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of EMNLP: System Demonstrations*, pages 66–71.

Hung Le and 1 others. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In *NeurIPS*.

Yujia Li and 1 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Long Ouyang and 1 others. 2022. Training language models to follow instructions with human feedback. *arXiv:2203.02155*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Rafael Rafailov and 1 others. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Baptiste Rozière and 1 others. 2023. Code llama: Open foundation models for code. *arXiv:2308.12950*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Nisan Stiennon and 1 others. 2020. Learning to summarize with human feedback. In *NeurIPS*.

The Unicode Consortium. 2024. Unicode standard annex #15: Unicode normalization forms. Version current at time of writing.

Edwin B. Wilson. 1927. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212.

Eric Zelikman, Yuhuai Wu, Noah D Goodman, and Tomer Holzman. 2022. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*.

PyBangla at BLP-2025 Task 2: Enhancing Bangla-to-Python Code Generation with Iterative Self-Correction and Multilingual Agents

Jahidul Islam, Md Ataullha, Saiful Azad
Department of Computer Science and Engineering
Green University of Bangladesh, Dhaka, Bangladesh
jahidul.cse.gub@gmail.com ataullha00@gmail.com saiful@cse.green.edu.bd

Abstract

LLMs excel at code generation from English prompts, but this progress has not extended to low-resource languages. This paper addresses the challenge of Bangla-to-Python code generation by introducing BanglaCodeAct, an agent-based framework that leverages multi-agent prompting and iterative self-correction. Unlike prior approaches that rely on task-specific fine-tuning, BanglaCodeAct employs an open-source multilingual LLM within a Thought–Code–Observation loop, enabling the system to dynamically generate, test, and refine code from Bangla instructions. We benchmark several prominent small-parameter open-source LLMs and evaluate their effectiveness on the mHumanEval dataset for Bangla NL2Code. Our results show that Qwen3-8B, when deployed with BanglaCodeAct, achieves the best performance, with a pass@1 accuracy of 94.0% on the development set and 71.6% on the blind test set. These findings establish a new benchmark for Bangla-to-Python translation and highlight the potential of agent-based reasoning for reliable code generation in low-resource languages.. Experimental scripts made publicly available at github.com/jahidulzaid/PyBanglaCodeActAgent

1 Introduction

Large Language Models (LLMs) has created a paradigm shift in software engineering, automating complex coding tasks, and democratizing programming for a larger audience (Chen et al., 2021). Natural Language-to-Code (NL-to-Code) generation (Yin et al., 2022), once a distant goal, is now a tangible reality, with systems capable of producing functional code from simple English descriptions. The vast majority of these advances remain linguistically monolithic, centered almost exclusively on English, leaving other languages behind.. This linguistic bias creates a significant accessibility gap for millions of learners worldwide whose primary language is not English. For speakers of low-resource languages like **Bangla**, the seventh most spoken language globally.

To address this critical issue, we introduce the **BanglaCodeAct Agent**, a ReAct agent framework designed for cross-lingual code generation from Bangla instructions into executable Python. Instead of relying on task-specific fine-tuning, our approach leverages the emergent multilingual reasoning capabilities of a general-purpose open-source LLMs within an iterative, self-correcting loop.

We address 3 research questions:

1. **RQ1:** How can an agent-based framework be designed to generate Python code from natural language instructions in Bangla?

2. **RQ2:** Can a general-purpose, multilingual LLMs be effectively prompted to perform cross-lingual code generation in a zero-shot setting, without the need for task-specific datasets?

3. **RQ3:** How does incorporating an iterative *Thought-Code-Observation* loop within a robust execution environment affect the reliability and correctness of the generated code?

2 Related Work

This research is positioned at the confluence of several rapidly advancing domains: automated code generation, the development of specialized large language models for programming, and the specific challenges within Natural Language Processing (NLP) for low-resource languages like Bangla. Our work synthesizes insights from these areas to address a novel problem: agent-driven,

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 566–570
December 23, 2025 ©2025 Association for Computational Linguistics

cross-lingual code generation from a low-resource language.

The introduction of the Transformer architecture (Vaswani et al., 2017) created major progress in this field. This led to the development of Large Language Models (LLMs) trained on vast web-scale corpora. Models like OpenAI's Codex, the engine behind GitHub Copilot (Chen et al., 2021), and DeepMind's AlphaCode (Li et al., 2022a), which achieved competitive performance in programming contests. However, a significant limitation of this era has been a reliance on English-centric data and evaluation benchmarks.

Building on the success of general-purpose LLMs, a new wave of models has been specifically trained or fine-tuned for programming tasks. Notable examples include CodeLlama (Roziere et al., 2023); StarCoder (Li et al., 2023); and DeepSeek Coder (Guo et al., 2024).

To reduce hallucination and improve factual grounding, Retrieval-Augmented Generation (RAG) retrieves relevant documents from an external knowledge base and supplies them as context to the LLMs(Lewis et al.). Corrective RAG (CRAG) introduces a lightweight retrieval evaluator to augment retrieved documents, improving the correctness of the generation process (Yan et al., 2024).

Bangla NLP faces persistent gaps that make NL2Code especially challenging. First, **data scarcity**: large-scale parallel corpora of Bangla programming instructions and code are virtually absent (Zhong et al., 2024; Raihan et al., 2025a). Second, **morphological complexity**: Bangla's rich inflectional system makes natural language instructions harder to parse into precise logical forms compared to English (Bhattacharjee et al., 2023). Prior LLM-based approaches, trained or fine-tuned primarily on English or multilingual data, often fail to capture these nuances, resulting in low accuracy and unstable performance in Bangla NL2Code tasks (Chen et al., 2021; Li et al., 2022b).

Our work addresses these gaps by introducing **BanglaCodeAct**, which directly leverages the multilingual reasoning abilities of general-purpose LLMs in a self-correcting loop, without requiring costly Bangla-specific fine-tuning or large annotated datasets.

3 Dataset and Evaluation Metrics

The task involves translating Bangla natural language programming instructions into Python code, ensuring functional correctness by passing associated test cases. This setup mirrors typical NL2Code challenges but places a specific emphasis on low-resource language understanding and algorithmic reasoning in Bangla (Raihan et al., 2025c). To evaluate this translation process, we employ the **mHumanEval dataset** (Raihan et al., 2025a), which is tailored for Bangla-to-Python code generation. The dataset consists of natural language programming problems in Bangla, each paired with a corresponding Python implementation and unit test cases that serve as an objective correctness signal. It covers a wide range of fundamental programming concepts, including algorithmic reasoning, control structures, data manipulation, and function design. The sample structure of the dataset is presented in Table 1.

3.1 Evaluation Metric

The primary metric for our evaluation is **pass@1** on the HumanEval benchmark (Raihan et al., 2025a). A generated code snippet is considered a "pass" if it executes without error and satisfies all provided assertions in the 'test_list' for that problem.

$$\text{pass@}k := E_{\text{problems}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right]$$

4 Methodology

In this work we introduce an agent framework, **BanglaCodeAct Agent**, for cross-lingual code generation. The primary objective is to translate natural language programming instructions articulated in a low-resource language, **Bangla (Bengali)**, into executable Python code. The methodology hinges on a powerful multilingual Large Language Models (LLMs) integrated into an iterative reasoning and self-correction loop, enabling it to bridge the semantic gap between Bangla prose and Python's formal syntax.

4.1 Models and Baselines

We compare the performance of our proposed **BanglaCodeAct Agent** against several baselines to evaluate the contribution of its components and to situate its performance relative to other

ID	Instruction (Bengali)	Test Cases
1	একটি ফাংশন লিখুন যা পরীক্ষা করবে প্রদত্ত স্ট্রিং প্যালিনড্রোম কিনা। খালি স্ট্রিংক প্যালিনড্রোম হিসেবে গণ্য হবে। Example: `is_palindrome(s)`	assert is_palindrome(“TENET”) == True assert is_palindrome(“Bangla”) == False assert is_palindrome(“ ”) == True
2	একটি ফাংশন লিখুন যা একটি স্ট্রিং-এর মধ্যে থাকা শব্দগুলোকে উল্টো করে সাজাবে। Example: `reverse_words(string)`	assert reverse_words(“hello”)==“hello” assert reverse_words(“ a b ”) == “b a” assert reverse_words(“hello world”) ==“world hello”
3	একটি পাইথন ফাংশন লিখুন যা দিয়ে দুইটি পূর্ণসংখ্যার বিপরীত চিহ্ন আছে কিনা তা পরীক্ষা করা যায়। Example: opposite_Signs(n1, n2)	assert opposite_Signs(1,-2) == True assert opposite_Signs(3,2) == False assert opposite_Signs(-10,-10) == False

Table 1: The dataset for Shared Task 2 (Code Generation) includes Bengali programming instructions, the corresponding Python code implementations, and test cases designed for validation.

approaches. First, **Zero-Shot Prompting** serves as a direct baseline where the model is given only the system prompt and the user task (Bangla instruction plus test cases) and is asked to generate the solution in a single turn. This approach achieves varying results across models: Qwen/Qwen3-8B obtains 36%, Qwen-Coder-7B reaches 51%, TigerLLM-1B-it (Raihan et al., 2025b) it achieves 11%, and Llama-3.1-8B performs the best at 77%. Next, **Few-Shot Prompting** provides the model with a small number of solved examples in the prompt to help it generalize to new problems. Performance here also varies, with Qwen3-8B achieving 46%, Qwen2.5-Coder-7B reaching 51%, and Llama-3.1-8B again performing strongly at 77%. DeepSeek-Coder-V2-Lite shows competitive results with a pass@1 of 73.0%. The **Self-Consistency** method leverages Qwen/Qwen3-8B to generate multiple independent solutions for the same problem and selects the final answer through majority voting, without using any iterative feedback loop. Finally, our full proposed framework, the **BanglaCodeAct Agent**, based on Qwen/Qwen3-8B, significantly outperforms these baselines with a 94% success rate. This agent employs an iterative *Thought-Code-Observation* loop, allowing it to self-correct based on execution feedback until all test cases are satisfied.

The experiments were executed with inference controlled by the hyperparameters presented in Table 2.

The agent's core is the **Qwen/Qwen3-8B** model, a multilingual LLM capable of zero-shot Bangla-to-logic translation and reasoning. To enable efficient multi-turn reasoning, we deploy it with the **vLLM inference engine**, leveraging **tensor parallelism** and **prefix caching** for reduced latency and high throughput (Kwon et al., 2023).

Parameters	**Value**
Max tokens	8192
Temperature	0.7
Top-p	0.9
Best-of	1
Repetition penalty	1.05 (CoT)
Decoding	Self-consistency ($n = 5$)
Num paths	16 / 5 (SC)
Seed	42
Timeout	5 Seconds
Retries	25

Table 2: Inference hyperparameters. These decoding and sampling parameters control output length, diversity, reproducibility, and error handling.

4.2 Cross-Lingual BanglaCodeAct Agent Framework

We employ the Code Acting (`CodeAct`) paradigm to structure the agent's problem-solving process. This approach transforms code generation from a single-shot task into a dynamic, multi-step dialogue between the agent and a code interpreter. The agent operates on a *Thought-Code-Observation* cycle (as illustrated in Fig. 1):

1. **Thought:** The agent generates an internal monologue, outlining its understanding and plan for the task in `<thought>`, showcasing reasoning in Bangla before code generation.

2. **Code Generation:** The agent produces Python code based on the plan, enclosed in `<code>` with test assertions for immediate self-verification.

3. **Execution and Feedback:** The code runs in a sandboxed **PythonREPL** with a timeout. Errors, like `TypeError`, provide feedback for **iterative self-correction**, refining the solution until valid or a max iteration is reached, with the result in `<answer>`.

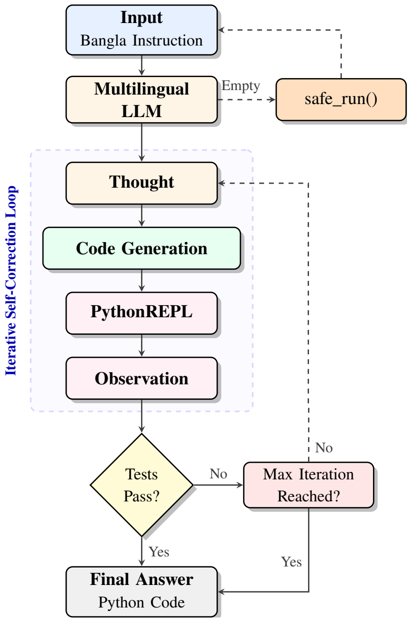


Figure 1: Thought-Code-Observation Cycle in the BanglaCodeAct Agent Framework. This diagram illustrates the iterative process of generating code, executing it, providing feedback, and refining the solution based on self-correction, facilitating cross-lingual code generation in Bangla.

To enhance reliability, we implement a **retry handler (`safe_run`)** that re-initiates the reasoning process if the agent produces an invalid or empty response. The retry mechanism permits a user-defined number of task attempts, improving success rates by reducing sporadic failures.

Instruction (Bengali)	**Test Cases**
স্ট্রিং থেকে প্রদত্ত অক্ষরের প্রথম এবং শেষ উপসর্গ মুছে ফেলুন। Example: `remove_Occ(s, ch)`	remove_Occ(”hello”,”l”) == ”heo” remove_Occ(”banana”,”a”) == ”bann” remove_Occ(”abc”,”x”) == ”abc”

একটি প্রদত্ত ম্যাট্রিক্সকে তার সারিগুলির যোগফল অনুযায়ী সাজান। Example: `sort_matrix(M)`	sort_matrix([[1,2,3],[2,4,5],[0,1,1]])
	== [[0,1,1],[1,2,3],[2,4,5]]
	sort_matrix([[5,5],[2,2],[3,3]])
	== [[2,2],[3,3],[5,5]]

Table 3: Illustrating error recovery in ambiguous and complex cases.

For instance, “স্ট্রিং থেকে প্রদত্ত অক্ষরের প্রথম এবং শেষ উপসর্গ মুছে ফেলুন” (remove the first and last occurrence of a given character from a string). Initial attempts produced incomplete logic (removing only one occurrence). (see Table 3).

5 Results and Analysis

Different models and experiments were conducted during the development phase, which are reported in 4.1. The experiment setup and hyperparameter details are described in table 2.

The ‘pass@1‘ scores for all evaluated methods on the mHumanEval dataset are summarized in Table 4. Our proposed BanglaCodeAct Agent achieves a ‘pass@1’ score of **94.0%**, significantly outperforming all other methods.

LLM Model	Method	pass@1
Qwen3-8B	**BanglaCodeAct**	**94.0**
Qwen3-8B	Self-Consistency	88.0
Qwen3-8B	Majority Voting	66.0
Qwen3-8B	Few-Shot	46.0
Qwen3-8B	Zero-Shot	36.0
Qwen2.5-Coder-7B	Few-Shot	51.0
Qwen2.5-Coder-7B	Zero-Shot	44.0
Llama-3.1-8B	Zero-Shot	39.0
Llama-3.1-8B	Few-Shot	77.0
DeepSeek-Coder-V2-Lite	**BanglaCodeAct**	**73.8**
DeepSeek-Coder-V2-Lite	Few-Shot	73.0
DeepSeek-Coder-V2-Lite	Zero-Shot	71.4
TigerLLM-1B-it	Zero-Shot	11.0

Table 4: Comparison of pass@1 accuracy (%) for different models and prompting strategies on the mHumanEval dataset. Our proposed **BanglaCodeAct Agent** (Qwen3-8B) achieves the highest score, demonstrating the effectiveness of iterative self-correction.

The results in Table 4, clearly demonstrate the efficacy of our agent-based framework.

The experimental results demonstrate the effectiveness of the proposed **BanglaCodeAct Agent** in leveraging an iterative self-correction mechanism for Bangla-to-Python code generation. With the Qwen3-8B model, the agent achieves a **94.0% pass@1 accuracy**, significantly outperforming Zero-Shot (36.0%), Few-Shot (46.0%), and Majority Voting (66.0%) strategies (Table 4). It ranked 17th on the test set (71.6%) and 8th on the development set (94%).

These results underscore the agent's ability to correct common code generation errors using REPL feedback, which distinguishes it from static prompting approaches. The Qwen3-8B model outperforms specialized models like Qwen2.5-Coder-7B, highlighting the importance of multilingual reasoning over code-specific training. Primary failure cases occur with semantically ambiguous or complex instructions, where the agent may not converge within 10 iterations.

6 Limitations

Despite strong performance, BanglaCodeAct has several limitations. The model's effectiveness is limited by its size, and experiments with larger LLMs (e.g., 32B parameters or more) were not conducted due to GPU resource constraints. Such models could potentially improve code generation accuracy for more complex tasks.

Additionally, the current evaluation primarily focuses on algorithmic and syntactic correctness. The system's ability to understand semantics and handle ambiguous or context-dependent Bangla instructions remains an open challenge. Moreover, the system relies on high-quality test cases for feedback, which may not always be available in real-world scenarios. The performance could be further limited by the absence of such reliable test cases in practice.

References

Abhishek Bhattacharjee, Shammur Aktar, Md. Saidul Islam Zishan, Sudipta Ghosh, Md. Mahadi Hasan, M. Sohel Rahman, Mohammad Ruhul Alam, Tetsuji Nakagawa, Teruhisa Misu, and Farig Shah. 2023. BanglaBERT: A Denoising Autoencoding based Pre-trained Language Model for Bengali. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2577–2591.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique P. de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, and 1 others. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *CoRR*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. Retrieval-augmented generation for knowledge-intensive nlp tasks.

R Li, LB Allal, Y Zi, N Muennighoff, D Kocetkov, C Mou, M Marone, C Akiki, J Li, J Chim, and 1 others. 2023. Starcoder: May the source be with you! *Transactions on machine learning research*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Dadashi, D O R A Lazic, Petar Veličković, Jiayuan Son, Johanni Botha, and et al. 2022a. Competition-Level Code Generation with AlphaCode. *Science*, 378(6624):1092–1097.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, and 1 others. 2022b. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Tworkowski, Marie-Anne Lachaux, Thibaut Lavril, Iz Mason, Alexandre Masson, Arthur Mensch, and 1 others. 2023. Code Llama: Open Foundation Models for Code. *arXiv preprint arXiv:2308.12950*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *Available at SSRN 5267341*.

Pengcheng Yin, Wen-Ding Li, Kefan Xiao, Abhishek Rao, Yeming Wen, Kensen Shi, Joshua Howland, Paige Bailey, Michele Catasta, Henryk Michalewski, and 1 others. 2022. Natural language to code generation in interactive data science notebooks. *arXiv preprint arXiv:2212.09248*.

Tianyang Zhong, Zhenyuan Yang, Zhengliang Liu, Ruidong Zhang, Yiheng Liu, Haiyang Sun, Yi Pan, Yiwei Li, Yifan Zhou, Hanqi Jiang, and 1 others. 2024. Opportunities and challenges of large language models for low-resource languages in humanities research. *arXiv preprint arXiv:2412.04497*.

Barrier Breakers at BLP-2025 Task 2: Enhancing LLM Code Generation Capabilities through Test-Driven Development and Code Interpreter

Sajed Jalil
Apollo Applications Group
Reston, VA, USA
sajed@apolloapps.ai

Shuvo Saha
University of Dhaka
Dhaka, Bangladesh
bsse0705@iit.du.ac.bd

Hossain Mohammad Seym
Islamic University of Technology
Dhaka, Bangladesh
hossainseym@iut-dhaka.edu

Abstract

Over the past few years, improving LLM code generation capabilities has been a key focus in NLP research. Despite Bengali having 242 million native speakers worldwide, it receives little attention when it comes to training LLMs. More recently, various fine-tuning and augmented generation techniques have been employed to significantly enhance code generation performance. However, they require considerable expertise and resources to utilize effectively as an end user. The goal of our work is to democratize access to powerful code generation tools in resource-constrained emerging markets, enabling users to leverage them in their native language.

We introduce a novel approach that combines Test-Driven Development (TDD) and Code Interpreter (CI), utilizing open-weight models, which improves the baseline accuracy for code generation with Bengali prompts and achieves an overall accuracy of **85%**. Our approach requires no finetuning and proves that even the smallest models in the same family can attain up to **98%** accuracy compared to the largest models. All of our results [1] are publicly shared in GitHub for validation and reproducibility.

1 Introduction

Large Language Models (LLMs) have gained significant attention across various research communities since the release of ChatGPT in 2022 [2]. Initially known as generalized text completion models, LLMs quickly found their way into more specialized tasks such as code, image, and audio generation. Specifically, the impact is visible in the code generation domain. There has been a significant transformation in the daily workflow of the software engineers with these models (Jalil, 2025).

Despite being the 5th most spoken language worldwide, Bengali is not included in most of the top models as a primary language for training data (Raihan et al., 2025b). Even in cross-lingual settings, most models tend to reflect Western perspectives (Myung et al., 2024). Additionally, prior studies have demonstrated that multilingual tokenizers are often inefficient and require additional resources during training (Ali et al., 2024).

With these constraints in mind, we propose our work on enhancing existing open-weight LLMs of various sizes by combining Test-Driven Development (TDD) and Code Interpreter (CI) without the need for fine-tuning. In this shared task with Bengali prompts, we investigate the following research questions that are crucial for advancing the field of multilingual code generation -

RQ1: *How far can performance improve without fine-tuning or external data augmentation?*

RQ2: *Can smaller models approach larger model performance?*

RQ3: *What approach is most effective in improving vanilla (baseline) LLM accuracy?*

RQ4: *To what extent do these approaches reduce compilation errors?*

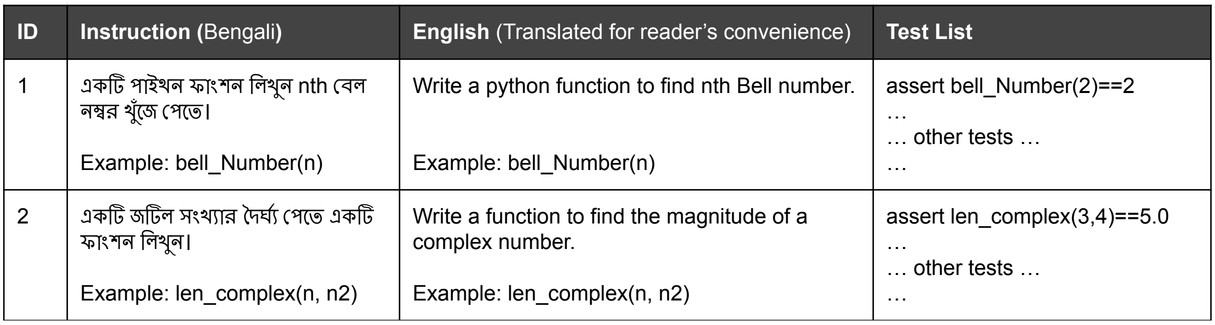
2 Background

Although there have been significant prior studies in NLG and benchmarks for Bengali (Bhattacharjee et al., 2022; Ekram et al., 2022; Raihan et al., 2025a), the number of code generation studies using LLM is quite negligible. The only substantial study we could find is a family of finetuned models named *TigerCoder*, which was evaluated for its machine translation capabilities (Raihan et al., 2025b). These findings underscore the need for further exploration using alternative techniques to improve code generation capabilities.

Test-Driven Development (TDD) has been a widely researched methodology in the agile soft-

[1] https://github.com/sajedjalil/BLP25-Task-2/
[2] https://chatgpt.com/

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 571–576
December 23, 2025 ©2025 Association for Computational Linguistics



ID	Instruction (Bengali)	English (Translated for reader’s convenience)	Test List
1	একটি পাইথন ফাংশন লিখুন nth বেল নম্বর খুঁজে পেতে। Example: bell_Number(n)	Write a python function to find nth Bell number. Example: bell_Number(n)	assert bell_Number(2)==2 … … other tests … …
2	একটি জটিল সংখ্যার দৈর্ঘ্য পেতে একটি ফাংশন লিখুন। Example: len_complex(n, n2)	Write a function to find the magnitude of a complex number. Example: len_complex(n, n2)	assert len_complex(3,4)==5.0 … … other tests … …

Figure 1: Example of dataset rows used in our study (English instruction is added here for readers’ convenience.)

ware engineering domain (Shull et al., 2010; Rafique and Mišić, 2012). It is the practice of writing unit tests before starting implementation to ensure software verification. This methodology has been proven to reduce code defects (Williams et al., 2003). To the best of our knowledge, no other prior studies have explored the effects of TDD in code generation with Bengali prompts.

Code Interpreter (CI) can act as an external tool to help LLM improve itself as a coding agent (Wang et al., 2024). Humans interact with LLM multiple times if the desired output is not reached (Lin et al., 2025). This inspired us to utilize CI to enhance accuracy and minimize compilation errors in our study. Additionally, we employed a combined approach that incorporates TDD and CI to improve accuracy further and reduce compilation errors.

3 Task Dataset

The primary aim of this task was to generate Python code from Bengali instructions using LLM (Raihan et al., 2025a,c,b). All of our code and experimental results are publicly available on GitHub. [3]

In Figure 1, a sample of the dataset is shown. The test cases evaluate the generated code. Only one test case was publicly available during the competition. The rest were hidden and could only be accessed after the submission phase had ended.

Model family	Used variants
Meta Llama 3.2	3B, 11B, 90B
Meta Llama 4	Scout 17B, Maverick 17B
OpenAI gpt-oss	20B, 120B

Table 1: Distribution of LLM models and variants in our experiment.

4 Experiments

Our LLM responses were generated with the AWS Bedrock platform [4]. Therefore, our selection of various models was dependent upon the availability in Bedrock. Specifically, we experimented with the following models in Table 1.

Our initiative focused on improving the accuracy of generalized LLM code generation without fine-tuning. To achieve this, we have experimented with the following five approaches -

4.1 Vanilla (Baseline) Model

To establish baseline accuracy with the Bengali instruction, we conducted this experiment with plain (vanilla) LLM API to determine how different LLMs perform.

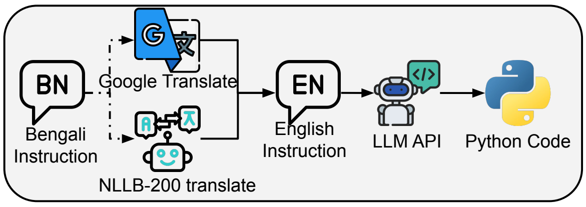


Figure 2: Two variants of Bengali to English machine translation.

4.2 Bengali to English Machine Translation

Since the primary language for most LLMs is English, our initial intuition was to translate the Bengali instructions into English. For this experiment, we have used two different translators - Google Translate [5] & NLLB-200 (Costa-Jussà et al., 2022). The overall workflow of this approach is displayed in Figure 2.

[3] https://github.com/sajedjalil/BLP25-Task-2/

[4] https://aws.amazon.com/bedrock/

[5] https://translate.google.com/

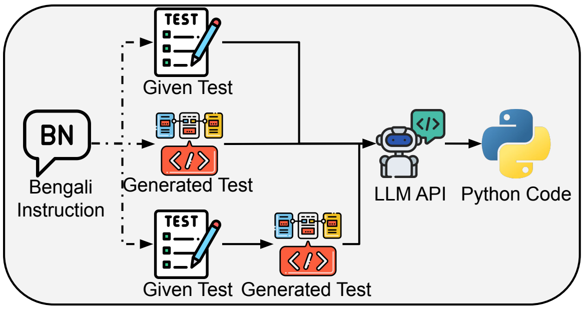


Figure 3: Variants of Test-Driven Development (TDD) approaches in our experiments.

4.3 Test-Driven Development (TDD)

We experimented with three different variations of TDD in this experiment. The detailed diagram is shown in Figure 3.

1. **Generated Tests** - We started with an API call to an LLM to generate up to five test cases from the given prompt. We then input these test cases, along with the given prompt, to generate our final response.

2. **Given Test** - We injected only the publicly available assert statement (test case) from the dataset into the LLM prompt during code generation.

3. **Combined** - This approach combined the above two methods. Here, we used the given test case from the dataset, along with five more LLM-generated test cases. And then, all of these test cases were used in LLM for response generation.

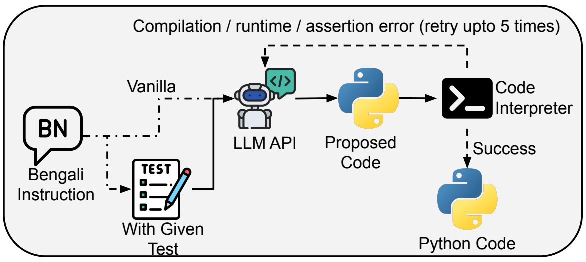


Figure 4: Code Interpreter with Test-Driven Development (TDD) approach.

4.4 Code Interpreter (CI)

We drew inspiration for this method from how developers interact with LLMs in real life. Developers generate code from LLM and then test the code in their respective IDE or environment. If any problem is encountered, they continue the chat and share error messages with the LLM until the desired output is achieved.

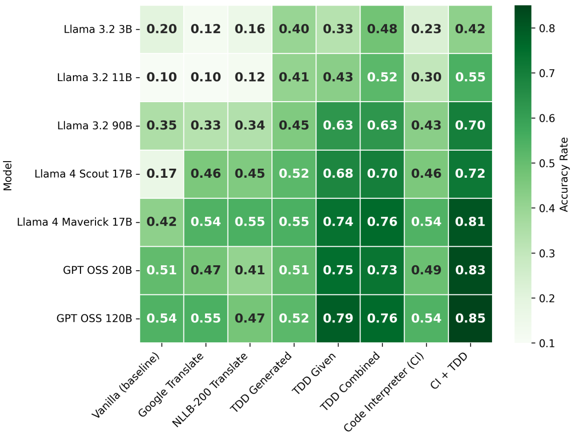


Figure 5: Overall accuracy heatmap of models in different approaches.

We utilized AWS Code Interpreter as a simulated Python environment [6]. Given a Python code, it compiled and executed it. For any errors, a detailed error message was obtained. We also set a retry limit of five to fix the generated Python code that did not compile. The error message received from the earlier execution was used as additional input for subsequent code generation. A workflow model is displayed in Figure 4 with the vanilla path.

4.5 CI+TDD

We combined the TDD approach with the dataset provided single test case with the CI. This test case was also executed in the interpreter to verify its success. The mechanism is demonstrated in Figure 4 with the given test path.

5 Results & Analysis

The results of our experiments are provided in Table 2 and Table 3. We group the data by model family and model parameters. The best outcome for each model is represented by bold text.

RQ1: ***How far can performance improve without fine-tuning or external data augmentation?***

In our investigation, we obtained several interesting findings. Figure 5 demonstrates the overall accuracy score on the test phase. It is distinctly evident from the heatmap that vanilla (baseline) accuracy can be improved significantly with TDD and CI. Except for the Llama 4 models, machine translation from Bengali to English did not provide

[6] https://docs.aws.amazon.com/bedrock/latest/userguide/agents-code-interpretation.html

Model	Vanilla	Translated		Test-Driven Development			Code Interpreter	
		Google	NLLB	Generated	Given	Combined	Vanilla	Given Test
Llama 3.2 models								
3B	19.6	12.4	16.0	39.6	33.4	**48.2**	22.6	42.2
11B	9.8	9.6	12.4	40.6	42.8	51.8	30.4	**54.8**
90B	35.0	32.8	34.2	44.8	62.8	63.4	42.6	**69.6**
Llama 4 models								
Scout	16.8	45.8	45.4	51.6	67.6	69.6	45.8	**72.0**
Maverick	42.0	54.0	54.6	54.6	74.4	76.4	54.4	**80.6**
GPT-OSS models								
20B	51.0	47.0	41.4	50.8	75.4	72.6	48.6	**82.8**
120B	54.4	54.6	46.8	52.2	79.0	75.6	54.0	**85.0**

Table 2: Overall accuracy (%) on varying model family and parameter size over different approaches.

Model	Vanilla	Translated		Test-Driven Development			Code Interpreter	
		Google	NLLB	Generated	Given	Combined	Vanilla	Given Test
Llama 3.2 models								
3B	21.4	61.8	38.4	8.4	7.6	5.8	**0.2**	0.8
11B	67.8	71.2	61.0	5.2	0.2	2.0	**0.0**	0.2
90B	8.8	15.4	9.4	**0.2**	**0.2**	0.4	0.4	**0.2**
Llama 4 models								
Scout	66.4	1.2	1.4	3.6	0.8	0.8	**0.2**	2.6
Maverick	19.2	0.2	0.2	1.2	**0.0**	0.2	0.2	**0.0**
GPT-OSS models								
20B	1.0	1.8	1.8	1.8	1.2	1.2	**0.2**	**0.2**
120B	0.2	0.4	0.2	0.4	0.8	1.2	**0.0**	0.2

Table 3: Overall compilation errors occurrence (%) on varying model family and parameter size.

significant improvement. Instead, it harmed the overall accuracy compared to the non-translation approach.

On the other hand, we observed an impressive increase in accuracy compared to the baseline in Figure 6. The CI+TDD approach improved the accuracy across all models by **+57%** to **+450%**. The TDD approach improves the baseline by **+47%** to **+420%**. Bengali to English machine translation has a change factor from **-20%** to **+171%**.

Compared to baseline (54%), overall accuracy can be improved up to 85% using our proposed techniques.

RQ2: ***Can smaller models approach larger model performance?***

In the Llama 3.2 model family, using the TDD 3B variant (48%) exceeds the baseline accuracy of the 90B variant (35%). Comparing the best outcome for each variant, we observed that 3B could reach **67%** of the performance of 90B and **87%** of the performance of 11B.

For the Llama 4 model family, Scout can exceed the Maverick baseline by 71% using CI+TDD. When comparing best outcomes, Scout can achieve up to **89%** of Maverick's performance.

Lastly, in the gpt-oss variants, the 20B variant using CI+TDD surpasses the 120B baseline by 54%. In best-case scenarios for both, 20B can reach up to **98%** of the performance of 120B. Our results further confirm the claims made by another prior study (Belcak et al., 2025).

In the same model family, the smallest model can attain up to 98% accuracy of the largest model.

RQ3: ***What approach is most effective in improving vanilla (baseline) LLM accuracy?***

As Figure 6 indicates that the best outcome is

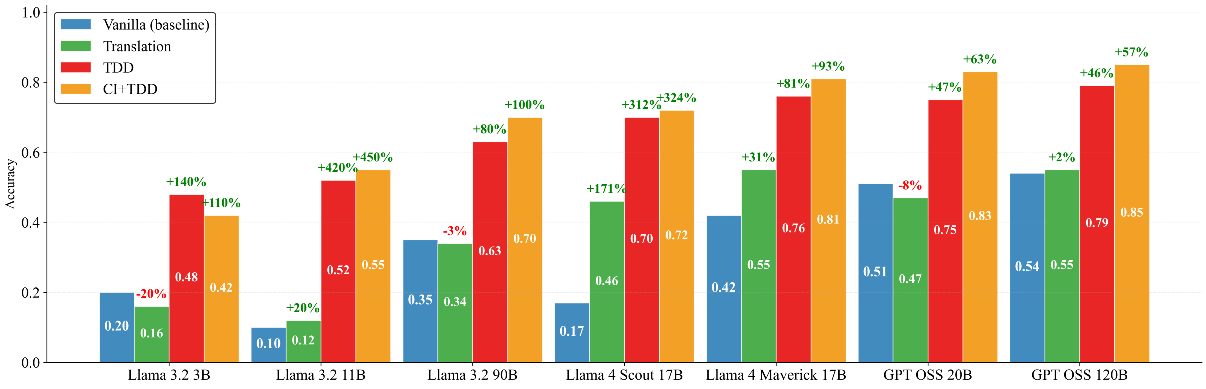


Figure 6: Model accuracy comparisons of baseline vs. our approaches (with increase/decrease in percentage).

always from CI+TDD except for the Llama 3.2 3B model. It should be noted that both TDD and CI+TDD performed significantly better than baseline in all models.

Combination of Test-Driven Development (TDD) and Code Interpreter (CI) yields the largest jump in accuracy.

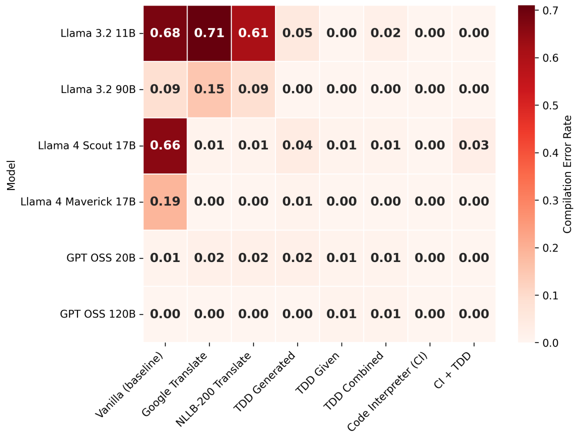


Figure 7: Compilation error rate heatmap of models in different approaches.

RQ4: ***To what extent do these approaches reduce compilation errors?***

In terms of total compilation errors in the generated code, a similar trend is visible as the accuracy rate. Figure 7 demonstrates both TDD and CI approaches have nearly eliminated all compilation errors with rates approaching **0%**. Translation helped reduce compilation errors only in the Scout model families. An interesting trend is observed in the gpt-oss model family, whose baseline compilation errors are nearly zero, suggesting it may contain inherent mechanisms to address compilation issues.

Both TDD and CI reduce compilation errors, whereas Bengali to English machine translation increases the error count in most cases.

Conclusion

This study successfully introduced a novel approach that combines **Test-Driven Development (TDD) and Code Interpreter (CI)** to improve code generation accuracy for Bengali prompts utilizing open-weight LLMs. Our findings demonstrate that this strategy yields significant improvements without requiring resource-intensive fine-tuning or the use of external data for augmentation. The **CI+TDD methodology** was the most effective, increasing overall baseline accuracy by up to 450% and virtually eliminating compilation errors across all models tested. Furthermore, our research suggests that using these strategies, even the smallest models in the same family can achieve up to **98% accuracy** when compared to the largest models of the same family. We strongly believe the exact mechanism can be applied to other underrepresented languages, similar to Bengali, and increase access to high-performing code generation tools in resource-constrained emerging markets.

6 Limitations

Our study is limited in context, as we only checked a subset of open-weight models available on the AWS Bedrock API. This restricted us from using several other popular models not available on that platform, such as Qwen3 and Gemma3. Moreover, we did not explore how our approach would perform in larger and complex coding tasks, as opposed to the single method-based problems provided in the shared-task dataset.

Acknowledgments

We greatly appreciate Apollo Applications Group [7] for granting us access to the AWS resources necessary to perform this study. And special thanks to Marium Binte Ibrahim Ema and Jeffrey Dershewitz for their reviews.

References

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, and 1 others. 2024. Tokenizer choice for llm training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, and Rifat Shahriyar. 2022. BanglaNLG and BanglaT5: Benchmarks and resources for evaluating low-resource natural language generation in bangla. *arXiv preprint arXiv:2205.11081*.

Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, and 1 others. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Syed Mohammed Sartaj Ekram, Adham Arik Rahman, Md Sajid Altaf, Mohammed Saidul Islam, Mehrab Mustafy Rahman, Md Mezbaur Rahman, Md Azam Hossain, and Abu Raihan Mostofa Kamal. 2022. Banglarqa: A benchmark dataset for under-resourced bangla language reading comprehension-based question answering with diverse question-answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2518–2532.

Sajed Jalil. 2025. The transformative influence of llms on software development & developer productivity. In *2025 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, pages 1–10.

Xinrui Lin, Heyan Huang, Kaihuang Huang, Xin Shu, and John Vines. 2025. Seeking inspiration through human-llm interaction. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Junho Myung, Nayeon Lee, Yi Zhou, Jiho Jin, Rifki Putri, Dimosthenis Antypas, Hsuvas Borkakoty, Eunsu Kim, Carla Perez-Almendros, Abinew Ali Ayele, and 1 others. 2024. Blend: A benchmark for llms on everyday knowledge in diverse cultures and languages. *Advances in Neural Information Processing Systems*, 37:78104–78146.

Yahya Rafique and Vojislav B Mišić. 2012. The effects of test-driven development on external quality and productivity: A meta-analysis. *IEEE Transactions on Software Engineering*, 39(6):835–856.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Forrest Shull, Grigori Melnik, Burak Turhan, Lucas Layman, Madeline Diep, and Hakan Erdogmus. 2010. What do we know about test-driven development? *IEEE software*, 27(6):16–19.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning*.

Laurie Williams, E Michael Maximilien, and Mladen Vouk. 2003. Test-driven development as a defect-reduction practice. In *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.*, pages 34–45. IEEE.

[7] https://www.apolloapps.ai/

Musafir at BLP_2025 Task 2: Generating Python Code from Bangla Prompts using a Multi-Model Cascade and Unit Test Validation

Sakibul Hasan[1], **Md Tasin Abdullah**[1], **Abdullah Al Mahmud**[1], **Ayesha Banu**[1]

[1]Department of Computer Science and Engineering,
Chittagong University of Engineering & Technology (CUET)

Correspondence:
u2004043@student.cuet.ac.bd, u2004059@student.cuet.ac.bd,
u2004057@student.cuet.ac.bd, ayesha.banu@cuet.ac.bd

Abstract

This paper presents our approach for the BLP25 Task 2: Code Generation in Bangla. To address the scarcity of Bangla–code training data, we adopt a two-stage pipeline. First, Bangla problem statements are translated into English using a neural translation model optimized for preserving technical semantics. Then, the translated text is passed to a Qwen-based code generation model to produce executable solutions. This translation–generation strategy leverages the strengths of English-centric code models while ensuring fidelity to the original Bangla instructions. Our system achieved competitive performance on the leaderboard, achieving the **3rd place** with a score of **91.8%** while demonstrating that translation-augmented pipelines are effective for low-resource code generation tasks.

Keywords: Large Language Model, Qwen, Unit Test.

1 Introduction

Natural language to code generation aims to translate human-readable descriptions into syntactically correct and semantically meaningful code. While notable advances have been achieved in English with large datasets and pretrained models, low-resource languages like Bangla remain underexplored. This limitation prevents native Bangla speakers from fully benefiting, as most existing code generation systems are not designed for Bangla instructions.

Bangla holds immense potential for broader accessibility in computational tasks. (Nahin et al., 2025) introduce TituLLMs, a family of Bangla LLMs with extensive benchmarking across diverse tasks. (Bhattacharjee et al., 2022) propose BanglaBERT, a pretrained language model and benchmark suite for low-resource Bangla understanding. (Kowsher et al., 2022) develop a transformer-based Bangla-BERT optimized for transfer learning and efficient Bangla language understanding. However, the lack of large, parallel Bangla–code datasets hinders the direct training and evaluation of Bangla code generation models. This gap shows the need for new strategies to adapt code generation systems to Bangla without large language-specific resources.

In this work, we propose a two-stage pipeline to address Bangla-to-code generation. First, Bangla problem descriptions are translated into English, enabling the use of pretrained English-centric code models. The translated instructions are then processed by a Qwen-based code generation model to produce executable solutions. This translation–generation framework mitigates the scarcity of Bangla–code data while leveraging multilingual modeling advances. Our system shows competitive performance, demonstrating translation as an effective bridge for low-resource code generation.

2 Related Work

Bangla is the seventh most spoken language in the world, yet it remains underrepresented in LLM research, particularly in code generation. The lack of Bangla–code parallel data has hindered systems that can natively process Bangla instructions. This gap has led to several recent initiatives to create Bangla-specific benchmarks and LLMs. (Yang et al., 2024) introduce Qwen2, the next-generation models in the Qwen family, ranging from 0.5B to 72B parameters in both dense and Mixture-of-Experts (MoE) variants. The report highlights improvements in multilingual coverage, reasoning ability, and training efficiency, with strong benchmark results across general language understanding, coding, and instruction-following tasks. Earlier, (Bai et al., 2023) presented the first Qwen technical report, outlining the development of large-scale multilingual models with specialized variants for chat, coding, and mathematics.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 577–581
December 23, 2025 ©2025 Association for Computational Linguistics

Their work established Qwen as a versatile family of open-source LLMs, demonstrating competitive performance against contemporaneous models and paving the way for subsequent expansions such as Qwen2.

(Raihan et al., 2025a) and BLP-2025 Task 2 (Raihan et al., 2025c) provide systematic benchmarks for evaluating Bangla code generation, revealing that existing multilingual models perform poorly on Bangla. Complementing these efforts, TigerCoder (Raihan et al., 2025b) introduces Bangla-specialized LLMs fine-tuned for code generation using a large instruction–code dataset built through translation and synthetic generation. Trained on the MBPP-Bangla benchmark, TigerCoder models outperform multilingual baselines, achieving state-of-the-art results in Bangla code generation. Our approach introduces a two-stage translation–generation pipeline where Bangla problem statements are translated into English and processed by a Qwen-based code model to generate Python solutions. Unlike TituLLMs and TigerCoder which trains Bangla-specialized LLMs on instruction–code datasets, our method leverages powerful English-centric models through translation to overcome data scarcity. Based on these benchmarks and models, we built our overall pipeline, shown in Figure 1, to effectively tackle Bangla-to-code generation.

3 Methodology

3.1 Dataset Description

Table 1: Dataset distribution across phases

Phase	No. of Samples in Dataset
Development phase	400
Testing phase	500

Each entry includes: Bangla natural language programming task and a set of assert statements or unit test cases used for designing, testing, and iteratively refining the pipeline.

3.2 Bangla to English Translation

The process for generating code from non-English instructions is illustrated in the accompanying figure 3. It begins with a Bangla Instruction, which is the input prompt. This instruction is processed by the Model Preparation stage, utilizing the gemma-3-12bit-unsloth-bnb model. In our approach, gemma-3-12bit-unsloth-bnb was used in a zero-shot inference setup with a custom prompt template, not fine-tuned. It served as a translation model to convert Bangla instructions into English while preserving technical semantics. Crucially, the model is provided with a Prompt Template to enforce the desired function structure and signature. Finally, the model's raw output is passed through the Translation stage, which decodes the model's output to produce the final, executable Python function. In short, the methodology shows how a language-specific prompt is combined with a structural template and processed by the Gemma model to perform a focused code generation task.

Table 2: Comparison of Bangla and English translations with verdicts.

Bangla Translation	English Translation	Verdict
একটি ত্রিভুজের পরিসীমা বের করার প্রোগ্রাম লিখ	Write a program to find perimeter of a triangle	Correct
একটি স্ট্রিং এর দীর্ঘতম পরবর্তী সাধারণ ক্রম বের করার প্রোগ্রাম লিখ	Write a program to find longest common string	Incorrect

3.3 LLM Inference & Code Generation

The translated English instruction, along with a specified code Prompt Template (defining the function signature), is fed into the Qwen 2.5-14B-Instruct model. This 14.7-billion-parameter LLM then performs the final, zero-shot code inference, producing the required Python function.

3.4 Unit Testing

The unit testing framework follows a structured workflow to ensure the correctness and reliability of generated code. As illustrated in Figure 4, the process begins with the Input stage, where predefined test cases are supplied as benchmarks for evaluating the code. These test cases are designed to cover a range of scenarios and edge cases, ensuring comprehensive assessment. In the Execution phase, the code is run within a dedicated, isolated environment, which guarantees secure and independent execution without interference from external processes. During the Testing stage, the outputs are parsed, validated, and compared against the expected results. The Evaluation phase further assesses consistency, correctness, and adherence to functional requirements, highlighting any deviations or unexpected behavior. Finally, the Decision stage categorizes each result as pass or fail, indicating whether the code satisfies the predefined

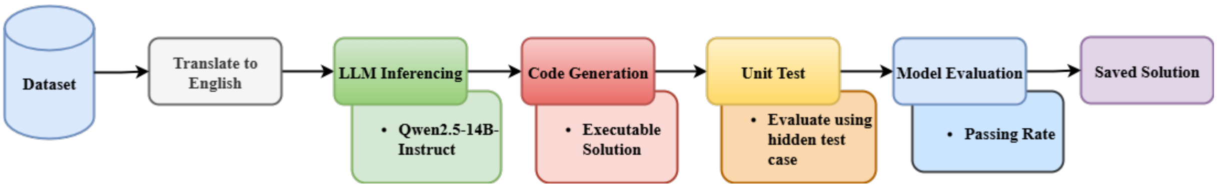


Figure 1: The overall pipeline describing the whole methodology covering the translation process to code generation with unit testing and model evaluation.

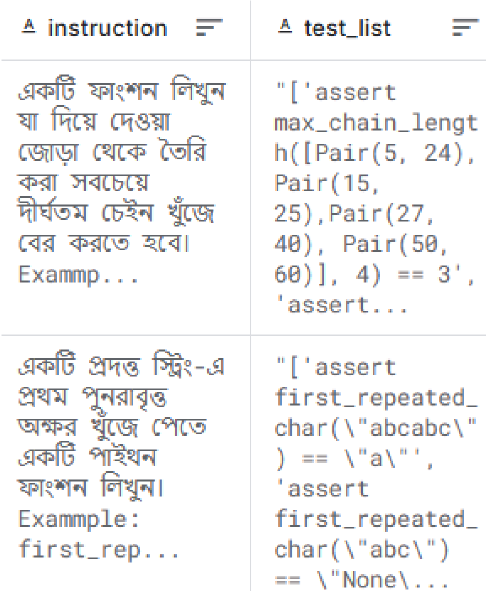
Table 3: Bangla and English prompts with corresponding code examples.

Bangla Instruction	English Instruction	Code	Result

একটি প্রদত্ত ম্যাট্রিক্সকে তার সারিগুলির যোগফল অনুযায়ী ক্রমবর্ধমান ক্রমে সাজানোর জন্য একটি ফাংশন লিখুন।	Write a function to sort a given matrix in ascending order based on the sum of its rows.	def sort_matrix(M):
		    return sorted(M, key=sum)
			Pass

একটি ত্রিভুজাকার প্রিজমের আয়তন খুঁজে বের করার জন্য একটি পাইথন ফাংশন লিখুন।	Write a Python function to find the volume of a triangular prism.	def find_Volume(l, b, h):
		    base_area = (l * b) / 2
		    volume = base_area * h
		    return volume
			Pass

প্রদত্ত সংখ্যাটি কাঠের বল কিনা তা পরীক্ষা করার জন্য একটি ফাংশন লিখুন।	Write a function to check if a given number is a Woodall number.	def is_woodall(x):
		    if x < 1:
		        return False
		    n = 1
		    while (2 ** n - 1) * n <= x:
		        if (2 ** n - 1) * n == x:
		            return True
		        n += 1
		    return False
			Fail



instruction	test_list
একটি ফাংশন লিখুন যা দিয়ে দেওয়া জোড়া থেকে তৈরি করা সবচেয়ে দীর্ঘতম চেইন খুঁজে বের করতে হবে। Exammp...	"['assert max_chain_length([Pair(5, 24), Pair(15, 25),Pair(27, 40), Pair(50, 60)], 4) == 3', 'assert...
একটি প্রদত্ত স্ট্রিং-এ প্রথম পুনরাবৃত্ত অক্ষর খুঁজে পেতে একটি পাইথন ফাংশন লিখুন। Exammple: first_rep...	"['assert first_repeated_char(\"abcabc\") == \"a\"', 'assert first_repeated_char(\"abc\") == \"None\...

Figure 2: Dataset Used in Development Phase

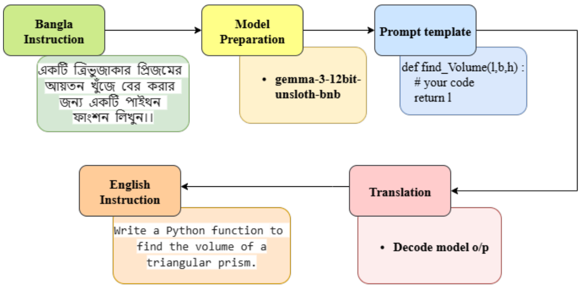

Bangla Instruction
একটি ত্রিভুজাকার প্রিজমের আয়তন খুঁজে বের করার জন্য একটি পাইথন ফাংশন লিখুন।।
Model Preparation
• gemma-3-12bit-unsloth-bnb
Prompt template
def find_Volume(l,b,h) :
# your code
return 1
Translation
• Decode model o/p
English Instruction
Write a Python function to find the volume of a triangular prism.

Figure 3: Bangla to English translation using gemma-3-12bit-unsloth-bnb model to generate the prompt in English.

requirements. Passing outcomes confirm that the solution meets the expected behavior, while failures reveal discrepancies that require correction. This structured approach provides a robust mechanism for automated verification, enabling reliable benchmarking of generated code in low-resource language scenarios.

4 Experiment & Analysis

There are two evaluation metrics for the task:

1. **Pass@1 (Passing rate)-** Number of hidden test cases that pass.

2. **Tie-breaker-** Shorter mean solution length.

Table 4: Comparison of **Pass@1** on different prompting methods between some Open-Source LLMs and our proposed solution.

Model	**Zero-Shot**	**Few-Shot**	**Instruction-Tuned**
Qwen2.5-Coder-14B-Instruct	82.7%	79.2%	81.4%
codegemma-7bit	76.8%	75.1%	75.7%
gpt-oss-20b	78.4%	76.9%	77.8%
Phi-4-reasoning	71.7%	67.8%	70.6%
CodeLlama-13b-Python	74.8%	72.3%	73.7%
Qwen2.5-14B + codegemma-7bit + gpt-oss-20B + phi-4-14B	**91.5%**	**88.5%**	**90.4%**

Table 5: Comparative Analysis of LLM Models: With vs Without Translation.

Model	**With Translation (%)**	**Without Translation (%)**
Combined	**91.5**	**87.5**
Qwen2.5-Coder-14B-Instruct	82.7	80.0
codegemma-7bit	78.4	75.0
Phi-4-reasoning	76.8	73.0

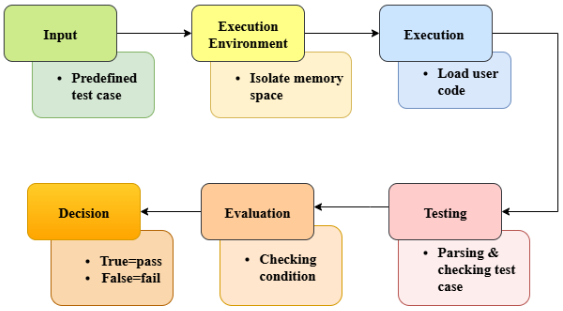


Figure 4: Unit testing framework to ensure the correctness and reliability of the code.

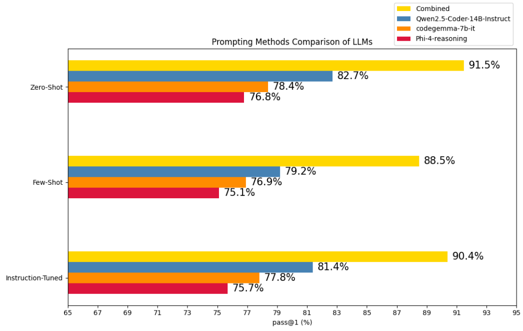


Figure 5: Comparative analysis on different prompting methods across models.

The table 4 compares the pass@1 of different models across three configurations: Zero-Shot, Few-Shot, and Instruction-Tuned. Among them Qwen2.5-14B, codegemma-7b, gpt-oss-20B, and phi-4-14B, achieves the highest performance, with an impressive 91.5% in Zero-Shot, 88.5% in Few-Shot, and 90.4% in Instruction-Tuned, demonstrating the effectiveness of the multi-model approach.

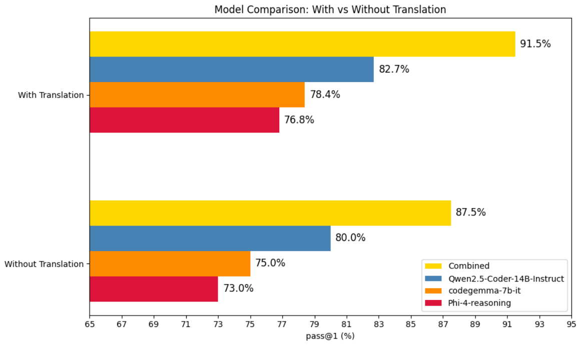


Figure 6: Comparative Analysis on Basis of Translation across Models.

The table 5 presents a comparative analysis of various LLM models with and without translation. The Combined model achieves the highest performance with 91.5% using translation, compared to 87.5% without it, showcasing benefits of translation. The results are also graphically represented in figure 5 & 6.

5 Conclusion

The paper presents an effective two-stage translation-generation pipeline to address the challenge of low-resource code generation in Bangla for the BLP25 Task 2. By first translating Bangla problem statements into English using a model optimized for technical semantics , and then passing the translated text to an English-centric Qwen-based code generation model , the approach successfully leverages the strengths of existing, powerful models designed for English. This strategy mitigates the scarcity of large, parallel

Bangla-code datasets and demonstrates that a translation-augmented framework is competitive for these tasks, achieving 3rd place with a score of 91.8% on the leaderboard.

Limitations

The proposed two-stage translation–generation pipeline shows promise for Bangla code generation but has limitations. It relies on English-centric models, which may miss technical nuances in Bangla prompts, and translation errors can affect output quality. Evaluation is limited to specific tasks, so testing on more diverse, real-world problems is needed to assess generalizability. [1]

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin 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 in bangla. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1318–1327.

M. Kowsher, Abdullah As Sami, Nusrat Jahan Prottasha, Mohammad Shamsul Arefin, Pranab Kumar Dhar, and Takeshi Koshiba. 2022. Bangla-bert: Transformer-based efficient model for transfer learning and language understanding. *IEEE Access*, 10:91855–91870.

Shahriar Kabir Nahin, Rabindra Nath Nandi, Sagor Sarker, Quazi Sarwar Muhtaseem, Md Kowsher, Apu Chandraw Shill, Md Ibrahim, Mehadi Hasan Menon, Tareq Al Muntasir, and Firoj Alam. 2025. Titullms: A family of bangla llms with comprehensive benchmarking. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24922–24940, Vienna, Austria.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

[1] Code is available here

JU_NLP at BLP-2025 Task 2: Leveraging Zero-Shot Prompting for Bangla Natural Language to Python Code Generation

Pritam Pal and **Dipankar Das**
Jadavpur University, Kolkata, India
{pritampal522, dipankar.dipnil2005}@gmail.com

Abstract

Code synthesis from natural language problem statements has recently gained popularity with the use of large language models (LLMs). Most of the available systems and benchmarks, however, are developed for English or other high-resource languages, and a gap exists for low-resource languages such as Bangla. Addressing the gap, the Bangla Language Processing (BLP) Workshop at AACL-IJCNLP 2025 featured a shared task on Bangla-to-Python code generation. Participants were asked to design systems that consume Bangla problem statements and generate executable Python programs. A benchmark data set of training, development, and test splits was provided, and evaluation utilized the Pass@1 metric through hidden test cases. We present here a system we developed, using the state-of-the-art LLMs through a zero-shot prompting setup. We report outcomes on several models, including variants of GPT-4 and Llama-4, and specify their relative strengths and weaknesses. Our best-performing system, based on GPT-4.1, achieved a Pass@1 score of 78.6% over the test dataset. We address the challenges of Bangla code generation, morphological richness, cross-lingual understanding, and functional correctness, and outline the potential for future work in multilingual program synthesis.

1 Introduction

The intersection of program synthesis and natural language processing has witnessed significant progress over recent years, with much of the work being done by large language models (LLMs). Code generation tasks, where a natural language expression of a problem is automatically converted into executable source code, have been explored extensively in English and specific high-resource languages (Lu et al., 2021; Chen et al., 2021; Austin et al., 2021). Very little work has been conducted on this task in low-resource languages, such as South Asian languages like Bangla (Raihan et al., 2025b), despite Bangla being one of the world's 10 most widely spoken languages and the national language in Bangladesh and second most widely spoken language in the Indian subcontinent.

To bridge this gap, a Shared Task on Code Generation for Bangla was organized as a task at the Bangla Language Processing (BLP) workshop at AACL-IJCNLP 2025 (Raihan et al., 2025c). The shared task entailed developing automatic systems to generate Python source code from Bangla problem statements. The organizers released a benchmark dataset with Bangla problem descriptions and corresponding Python solutions. The participants were asked to develop models/pipelines with the capacity to, upon being given a problem statement in Bangla, generate correct and executable Python programs.

This task is particularly challenging for several reasons. First, Bangla is a resource-scarce and morphologically dense language; thus, there are limited large-scale annotated data resources available for training. Second, we aim to bridge cross-lingual semantic understanding (natural language in Bangla) and code generation at its formal level (in Python), which can result in translation errors leading to syntactic or logical execution failures. Third, creating measures that capture more than surface-level textual similarity and move towards the functional correctness of code generation presents an additional level of challenge.

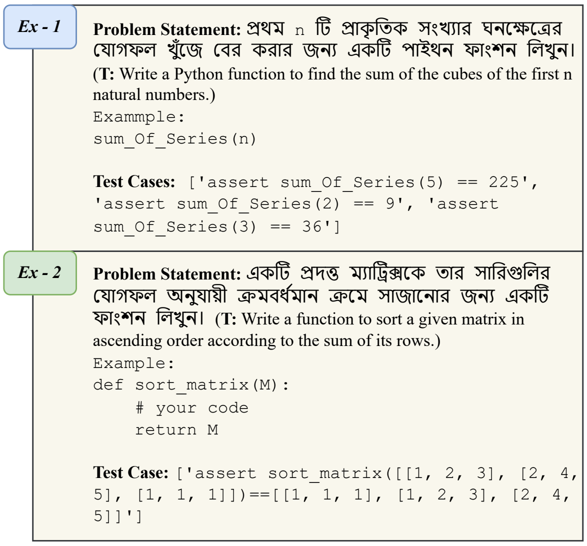
The shared task provides to the community a standardized dataset, evaluation framework, and collection of baseline results to promote orderly progress in this new field. Beyond benchmarking, it is desirable to stimulate the development of multilingual and cross-lingual code generation models and to support broader efforts to make programming more inclusive for speakers of Bangla and other learners and programmers.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 582–586
December 23, 2025 ©2025 Association for Computational Linguistics

2 Dataset

The datasets provided by the BLP-2025 Task 2 shared task organizers were used to develop the code generation framework. The organizers first provided a trial dataset to help understand the task and the input/output formats. Next, a development dataset (Raihan et al., 2025a) was provided by the organizers for the code generation framework, and a test dataset to evaluate its performance. The trial dataset comprises 74 problem statements in the Bengali language, each accompanied by corresponding Python source code and three executable test cases.

In the development dataset, there are a total of 400 problem statements in Bengali, with three test cases provided for each statement. The test dataset comprises 500 distinct problem statements in Bengali, which are entirely separate from those in the development dataset. This dataset is used to assess the performance of the code generation framework. To increase the challenge, the organizers only provided one test case for each problem statement in the test dataset, as opposed to the three provided in the development dataset. An example of the data from both the development and test datasets is shown in Figure 1.



Ex - 1

Problem Statement: প্রথম n টি প্রাকৃতিক সংখ্যার ঘনক্ষেত্রের যোগফল খুঁজে বের করার জন্য একটি পাইথন ফাংশন লিখুন। (**T:** Write a Python function to find the sum of the cubes of the first n natural numbers.)

```
Exammple:
sum_Of_Series(n)
```

Test Cases: `['assert sum_Of_Series(5) == 225', 'assert sum_Of_Series(2) == 9', 'assert sum_Of_Series(3) == 36']`

Ex - 2

Problem Statement: একটি প্রদত্ত ম্যাট্রিক্সকে তার সারিগুলির যোগফল অনুযায়ী ক্রমবর্ধমান ক্রমে সাজানোর জন্য একটি ফাংশন লিখুন। (**T:** Write a function to sort a given matrix in ascending order according to the sum of its rows.)

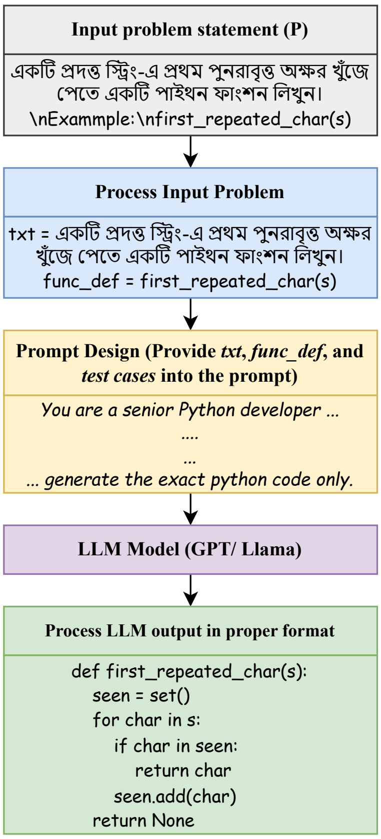
```
Example:
def sort_matrix(M):
    # your code
    return M
```

Test Case: `['assert sort_matrix([[1, 2, 3], [2, 4, 5], [1, 1, 1]])==[[1, 1, 1], [1, 2, 3], [2, 4, 5]]']`

Figure 1: Example of development data (Ex-1) and test data (Ex-2). The development data contains three test cases, while the test data contains only one test case.

3 Methodology

This section provides a brief overview of the overall methodology for code generation in the Bengali language. To develop the framework, we utilized state-of-the-art LLMs with a zero-shot setting.


Input problem statement (P)
Process Input Problem
Prompt Design (Provide *txt*, *func_def*, and *test cases* into the prompt)
LLM Model (GPT/ Llama)
Process LLM output in proper format

একটি প্রদত্ত স্ট্রিং-এ প্রথম পুনরাবৃত্ত অক্ষর খুঁজে পেতে একটি পাইথন ফাংশন লিখুন। \nExammple:\nfirst_repeated_char(s)

txt = একটি প্রদত্ত স্ট্রিং-এ প্রথম পুনরাবৃত্ত অক্ষর খুঁজে পেতে একটি পাইথন ফাংশন লিখুন।
func_def = first_repeated_char(s)

You are a senior Python developer ...
....
...
... generate the exact python code only.

```
def first_repeated_char(s):
    seen = set()
    for char in s:
        if char in seen:
            return char
        seen.add(char)
    return None
```

Figure 2: Overall framework of our code generation system. The pipeline begins with processing the Bangla problem statement and function definition, followed by carefully designed prompts. These prompts are then passed to an LLM, and the generated output is post-processed into valid Python code.

Task Definition: Given a problem statement P in the Bengali language, and test cases $T = \{t_1, t_2, ..., t_k\}$, where k is the number of test cases. Our primary objective is to develop a pipeline or framework that can accurately understand the Bengali problem statement and generate Python code that satisfies the test cases T.

Framework Description: The overall flow diagram of the framework is illustrated in Figure 2. Given its state-of-the-art performance across various tasks, such as machine translation, text summarization, sentiment analysis, and many others, we utilized LLMs, including `GPT-4.x` (OpenAI et al., 2024) and `Llama-4` (Touvron et al., 2023), as the core of the framework.

In Figure 2, the first step involves processing the input data provided by the organizers, which consists of two components. The first component is the problem statement, written in Bengali, followed by the text "\nExample\n", and then the function definition, which includes the function name and parameter list. This process is detailed in the 'Input Problem Statement' section of Figure 2.

We begin by processing the input data and storing it in two separate variables: one for the actual problem statement and the other for the function definition, along with its corresponding parameter list. This will facilitate further processing and prompt design.

The next phase involves designing the prompt. Since LLMs are trained on a vast amount of data and are capable of performing many complex NLP tasks, we employed the zero-shot prompting strategy (Brown et al., 2020; Kojima et al., 2022), where no proper example of the input-output structure is provided in the prompt. Each prompt was designed to include the problem statement (in the original Bengali text), the function definition, and parameter list, allowing for the development of the desired function. Along with the problem statement and function definition, the test cases were provided in the prompt so that the LLM could evaluate them to check whether it correctly generated the desired code. The prompt that was provided to the LLM models to generate the Python source code is provided in Figure 3.

Each LLM model was accessed with its corresponding API keys. During the generation of Python codes, the `temperature` value, `top_p` value, and maximum token limit were set to 0.2, 1, and 1024, respectively, across all LLM models.

Next, the generated Python code outputs were further processed to remove common code-block markdown markers (eg, "``` Python" or "```"). An example of generated Python code from a given problem statement is provided in Figure 2.

4 Experiment and Result

4.1 Experimental Setup

All experiments were conducted in the Google Colaboratory environment using a non-GPU virtual system with 12.7 GB of RAM. The GPT models were accessed via OpenAI's official website [1], while the Llama-4 model was obtained from a third-party provider, 'Together.AI' [2], utilizing the appropriate API keys. The experiments were set up in two configurations: one in which the LLM models were instructed to translate the provided Bengali problem statement into English (Figure 3), and another in which the problem statements were left in their original Bengali form [3].

> You are a senior Python developer. Your task is to write a single, well-structured Python function that solves the user's problem.
>
> Your output must contain only the code, with no conversational text. Only generate the proper executable code.
>
> It is to be noted that the problem statement is given in the Bengali language. You can translate the problem statement into English before writing the code.
>
> Problem: {txt}.
>
> The function definition, return type, and structure of code should be as follows: {func_def}
>
> There are test cases given in the program. Note that the generated code should pass the given test cases.
>
> Test Cases: {test_cases}
>
> Note that there might be other test cases also, along with the given test cases. Your solution should also be robust enough to handle other hidden test cases.
>
> Don't print the test cases. Only generate the exact Python code.

Figure 3: Prompt that was provided to LLMs for generating Python source code from Bengali problem statements. In this prompt, we instructed LLMs to translate the Bengali problem statement to English.

In order to evaluate the framework's performance, the Pass@1 metric was used, which is the percentage of hidden test cases that the generated Python code passes. The experiments were performed using different versions of GPT models, including `GPT-4-mini`, `GPT-4.1-mini`, `GPT-4.1`, and the `Llama-4`[4] model. To ensure a fair comparison, all the LLM models in both experimental setups were executed with the same `temperature`, `top_p`, and maximum token value as described in Section 3.

4.2 Result

The results for the test dataset are presented in Table 1 for both the translated and original Bengali

[1] https://openai.com/

[2] https://www.together.ai/

[3] Prompt for this experimental setup is provided in Appendix A

[4] Exact name is: `Llama-4-Maverick-17B-128E-Instruct-FP8`

LLM Model	Pass@1 (%)	Is translated to English
GPT 4o-mini	72.0	✓
GPT 4.1-mini	76.0	✓
GPT-4.1	**76.6**	✓
Llama-4	76.4	✓
GPT 4o-mini	65.6	✗
GPT-4.1-mini	74.0	✗
GPT-4.1	**78.6**	✗
Llama-4	75.6	✗

Table 1: Performance of different LLMs on the shared task test dataset, measured using the Pass@1 metric. The result highlighted with **Blue** colour represents the best performance when translating the Bengali problem statement into English (Official shared task submission). The result highlighted in **Green** colour represents the best performance when using original Bengali problem statements.

problem statement forms. From the table, it is evident that the `GPT-4.1` model performs best across both schemes, achieving Pass@1 scores of 76.6% with the translated problem statement and 78.6% with the original Bengali problem statement. The `Llama-4` model follows closely behind, obtaining a Pass@1 score of 76.4% for the translated form and 78.6% for the original form.

Notably, the `GPT-4.1` model demonstrates superior performance with the original Bengali problem statement compared to the translated version. This indicates a strong understanding of the Bengali language for code generation by the `GPT-4.1` model. In contrast, other models such as `GPT-4o-mini`, `GPT-4.1-mini`, and `Llama-4` performed better with the translated problem statement and showed decreased performance when using the original Bengali problem statement.

5 Conclusion and Future Work

This paper presents an LLM-based framework for Python code generation from Bengali problem statements. We used a zero-shot prompting strategy with three variants of GPT models and the Llama 4 model across two experimental setups: one translating the Bengali problem to English and another keeping the Bengali problem statement in its original form. All experiments were conducted on the development and testing datasets provided by the BLP Task 2 organizers. Our experimental results demonstrate that the GPT-4.1 model-based framework outperforms all other frameworks in both experimental setups, particularly when using the Bengali original problem statements, which achieved the best result with a 78.6% pass@1 score.

Future directions would include experimenting with other prompting strategies, such as few-shot or chain-of-thought prompting, and making a comparative analysis between different prompting strategies. Additionally, we will also experiment with other LLM models, such as DeepSeek (DeepSeek-AI et al., 2025) or Gemini (Team et al., 2025), in our future work.

Limitations

Our proposed work does have some potential limitations. First, we only experimented with a zero-shot prompting strategy and did not explore other prompting techniques, such as few-shot learning or chain-of-thought prompting. The performance of our approach may be enhanced by using these advanced strategies.

Second, our experiments were limited to only the GPT and Llama-4 models. We chose these models due to their relatively strong performance across various NLP tasks. However, we have not explored other LLMs, such as DeepSeek, Qwen, or Claude. Financial constraints have prevented us from doing so, as each model incurs costs for API calls.

Third, we did not conduct any explicit training or fine-tuning due to resource limitations. The performance could potentially improve if we fine-tune an LLM model, especially for the code generation task. We plan to attempt fine-tuning an LLM model for code generation in our future work.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 15 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, and 3 others. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *Preprint*, arXiv:2102.04664.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 9 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

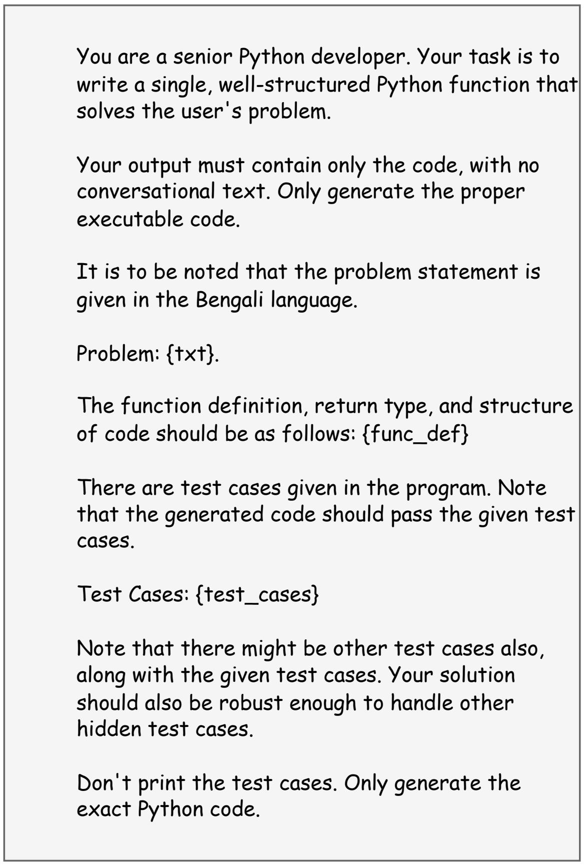
Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 5 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

A Appendix A: Prompt where no translation was performed



You are a senior Python developer. Your task is to write a single, well-structured Python function that solves the user's problem.

Your output must contain only the code, with no conversational text. Only generate the proper executable code.

It is to be noted that the problem statement is given in the Bengali language.

Problem: {txt}.

The function definition, return type, and structure of code should be as follows: {func_def}

There are test cases given in the program. Note that the generated code should pass the given test cases.

Test Cases: {test_cases}

Note that there might be other test cases also, along with the given test cases. Your solution should also be robust enough to handle other hidden test cases.

Don't print the test cases. Only generate the exact Python code.

Figure A.1: Prompt to Bengali-to-Python code generation where the problem statement was kept in its original Bengali form. No translation was conducted.

Retriv at BLP-2025 Task 2: Test-Driven Feedback-Guided Framework for Bangla-to-Python Code Generation

K M Nafi Asib, Sourav Saha, and Mohammed Moshiul Hoque
Department of Computer Science and Engineering
Chittagong University of Engineering & Technology, Chittagong 4349, Bangladesh
{nafi.asib, sahasourav1170}@gmail.com; moshiul_240@cuet.ac.bd

Abstract

Large Language Models (LLMs) have advanced the automated generation of code from natural language prompts. However, low-resource languages (LRLs) like Bangla remain underrepresented due to the limited availability of instruction-to-code datasets and evaluation benchmarks. To address this, the BLP Workshop at IJCNLP-AACL 2025 introduced a shared task on “Code Generation in Bangla”. In this work, we propose a method that combines instruction prompting with a test-driven, feedback-guided iterative refinement process using a fine-tuned Qwen2.5-14B model. The model generates code from Bangla instructions, tests it against unit tests, and iteratively refines any failing outputs through three evaluation passes, using test feedback to guide each step. This approach helped our team “Retriv” to secure 2nd place in the shared task with a Pass@1 score of 0.934. The analysis highlights challenges in Bangla instruction understanding and Python code generation, emphasizing the need for targeted methods in LRLs. We made experimental scripts publicly available for the community.[1]

1 Introduction

Automated code generation from natural language has made significant progress with Large Language Models (LLMs), which generate code snippets tailored to meet user needs. These models, trained on millions of open-source code repositories, perform best in high-resource languages such as English, where large, aligned datasets and benchmarks are available. In contrast, low-resource languages such as Bangla have received less attention due to the lack of high-quality instruction-to-code datasets (Raihan et al., 2025a). Recent work, including the TigerCoder models (Raihan et al., 2025b) and benchmarks such as mHumanEval (Raihan et al., 2025a) and MBPP-Bangla (Raihan et al., 2025b), has begun to address this gap by providing standardized datasets and evaluation protocols. Automated code generation in Bengali is important because it enables more people to access programming tools and resources in their native language, thereby supporting education and local software development.

To advance research, the BLP Workshop[2] at IJCNLP-AACL 2025 introduced a shared task on “Code Generation in Bangla” (Raihan et al., 2025c), providing a benchmark for evaluating models on Bangla instruction-to-Python code generation. This paper contributes to ongoing research, and the key contributions are as follows:

- Proposed a lightweight and effective system for Bangla-to-Python code generation that combines QLoRA fine-tuning with a feedback-guided inference loop and includes a test-case-aware translation step to ensure semantic alignment with expected input-output behavior.
- Conducted a systematic evaluation of several open-weight models, including (`ReasonFlux-Coder-14B`, `phi-4`, `Llama-3.1-8B`, `Qwen3-14B`, `Qwen2.5-Coder-14B`) on Bangla-to-Python code generation.

2 Related Work

Recent studies have sought to enhance the robustness of code generation. Python Code Generation by Asking Clarification Questions (Li et al., 2023) allows models to query ambiguous prompts, improving correctness through interactive clarification. Self-Debugging (Chen et al., 2023) proposes a general framework where LLMs iteratively refine their own outputs by leveraging execution feedback, showing improvements across

[1] `https://github.com/NafiAsib/Retriv-BLP25-Task-2`

[2] `https://blp-workshop.github.io/`

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 587–594
December 23, 2025 ©2025 Association for Computational Linguistics

text-to-SQL, code translation, and text-to-Python tasks. Similarly, the Large Language Model Debugger (LDB) (Zhong et al., 2024) incorporates runtime execution signals and block-level debugging, yielding gains on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). Building upon recent work in enhancing code generation, efforts to assess LLM capabilities in Bangla have led to the introduction of several benchmarks and resources. For example, the mHumanEval-Bangla dataset (Raihan et al., 2025a) extends HumanEval (Chen et al., 2021) to Bangla, while the MBPP-Bangla benchmark (Raihan et al., 2025b) adapts MBPP with crowd-sourced Bangla instructions paired with Python solutions. In parallel, modeling advances include TigerLLM (Raihan and Zampieri, 2025), a family of Bangla LLMs outperforming previous open alternatives, and TituLLMs (Nahin et al., 2025), which release pretrained Bangla models at 1B and 3B scales with comprehensive benchmarking.

Despite the availability of these benchmarks and improvements, code generation in Bangla remains relatively underexplored. The TigerCoder suite (Raihan et al., 2025b) is an early response to this gap, introducing Bangla-focused multilingual models and benchmarks. This work specifically addresses the task of Bangla-to-Python code generation by proposing a test-driven, feedback-guided refinement approach.

3 Task and Dataset Descriptions

The BLP Shared Task-2 tackled the challenge of developing robust Bangla code generation systems. Organizers provided instruction-to-Python code datasets for training (74 samples), development (400 samples), and testing (500 samples). Table 1 summarizes the structure and fields of each sample.

Field	Description
id	A unique task identifier.
instruction	A Bangla description of the programming task.
response	A Python code snippet implementing the task (available only in the training set).
test_list	A list of Python assert statements used for verifying functional correctness during development.

Table 1: Structure of each dataset sample.

4 System Description

Figure 1 illustrates the overview of our proposed framework.

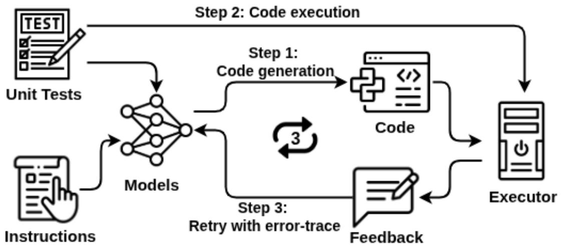


Figure 1: Overview of the proposed framework

4.1 Base Model Benchmarking

We began by comparing a diverse set of open-weight large language models (LLMs), chosen for their strong performance in code generation and multilingual understanding: `ReasonFlux-Coder-14B` (Wang et al., 2025), `phi-4` (Abdin et al., 2024), `Llama-3.1-8B` (Dubey et al., 2024), `codegemma-7b` (Team et al., 2024), `Qwen3-14B` (Yang et al., 2025), and `Qwen2.5-Coder-14B` (Hui et al., 2024). Each model was tested in a two-shot setting, where we prompted with two Bangla instructions and their reference solutions. To evaluate robustness, we repeated the same experiments using translated English instructions. Across both setups, `Qwen2.5-Coder-14B` consistently produced the most reliable and executable Python code, making it the natural choice for our subsequent fine-tuning.

4.2 Instruction Translation Strategy

Since the development dataset contained Bangla instructions paired with English test cases, we introduced an LLM-based translation step to better leverage English-centric code models. For each instruction, the full test suite was included in the prompt, allowing the translator to align the natural language description with the intended input/output behavior. This design helped preserve semantic fidelity, especially in tasks where the Bangla phrasing might otherwise be ambiguous. We used `Qwen2.5-Coder-14B` as the translator for the full dataset.

4.3 Fine-tuning with QLoRA

Given GPU memory constraints, we fine-tuned the base model using Quantized Low-Rank Adaptation (QLoRA) (Dettmers et al., 2023), which combines 4-bit weight quantization with low-rank adapters. This approach enabled us to adapt a 14B-parameter model within our compute budget while still leveraging its rich pretraining. For fine-tuning, we additionally used the DeepMind MBPP dataset (Austin et al., 2021) to supplement our training data. We settled on a configuration with rank $r = 128$, scaling factor $\alpha = 128$, a learning rate of 2×10^{-4}, and a batch size of 1 with gradient accumulation steps of 4. The training was performed for 4 epochs with weight decay of 0.

4.4 Feedback-guided Inference

Finally, we introduced a feedback mechanism to improve inference robustness. After the model generated a candidate solution, we executed it against all the test cases provided with a 30-second timeout. If any test failed, we re-prompted the model with the error trace, allowing it to iteratively refine its output. This loop was repeated up to three times, gradually increasing the sampling temperature ($0.1 \rightarrow 0.3 \rightarrow 0.5$) to encourage diverse candidate solutions. We found prompt design to be crucial here, and iteratively refined the error-handling prompt over multiple development runs.

5 Experiments

All experiments were conducted on a single NVIDIA RTX 3090 Ti GPU with 24 GB of memory. We used the HuggingFace[3] `transformers` library together with the `PEFT` framework and the `Unsloth`[4] package for efficient training. Unless otherwise stated, decoding was performed with a maximum of 768 tokens and an initial temperature of 0.1. This setup reflects a modest compute budget, which influenced the design of our training and inference strategies.

6 Results and Analysis

System performance was measured using the Pass@1 metric, defined as the proportion of top-1 generated solutions that pass all hidden unit tests. Leaderboard ranking was determined by Pass@1.

6.1 Few-shot on Bangla Instructions

We first evaluated all candidate models in a two-shot setting using the original Bangla instructions. As shown in Table 2, `Qwen2.5-Coder-14B` and `Qwen3-14B` emerged as the strongest models, reflecting their multilingual pretraining. In contrast, models with weaker multilingual grounding (`phi-4`, `codegemma-7b`) struggled to interpret Bangla instructions reliably.

Model	**Pass@1 (Bangla)**
ReasonFlux-Coder-14B	0.64
phi-4	0.20
Llama-3.1-8B	0.51
codegemma-7b	0.37
Qwen3-14B	0.67
Qwen2.5-Coder-14B	**0.74**

Table 2: Few-shot performance on Bangla instructions.

6.2 Few-shot on Translated Instructions

Next, we repeated the evaluation after translating Bangla instructions into English using our LLM-based pipeline. Table 4 shows that translation substantially boosted performance for most models, with `Qwen2.5-Coder-14B` again leading at 0.81 Pass@1. This demonstrates that while multilingual ability helps, aligning with English-centric pretraining remains advantageous for code generation. Table 3 illustrates representative Bangla instructions from the dataset alongside their English translations used for code generation.

6.3 Effect of Fine-tuning

Fine-tuning `Qwen2.5-Coder-14B` with QLoRA yielded a notable improvement, raising Pass@1 from 0.81 to 0.90. The best configuration was obtained with rank $r = 128$, scaling factor $\alpha = 128$, and 0 dropout. We found that any added dropout degraded performance, suggesting that preserving the full training signal was more important than regularization under the limited dataset size. This highlights the value of higher adapter ranks in capturing nuanced mappings from translated instructions to executable logic.

6.4 Feedback-guided Inference

Incorporating the feedback mechanism further improved performance to 0.94 Pass@1 (Table 5). Relative to the few-shot baseline, this represents a +16.05% absolute improvement, and a +4.44% gain over fine-tuning alone.

Most recoveries came from correcting off-by-one errors, handling edge cases, or aligning outputs with expected formats. However, failures

[3] https://huggingface.co/

[4] https://unsloth.ai/

Bangla Instruction	English Translation
প্রদত্ত পর্যায়ক্রমিক ফাংশনের জন্য সর্বনিম্ন সম্ভাব্য মান খুঁজে পেতে একটি পাইথন ফাংশন লিখুন।	Write a Python function to find the minimum possible value for a given arithmetic sequence.
প্রদত্ত তালিকায় টুপল বৈশিষ্ট্য হিসাবে রেকর্ড তালিকার সর্বাধিক মান খুঁজে পেতে একটি ফাংশন লিখুন।	Write a function to find the maximum value in each tuple's list within a given list and return the tuples with their respective maximum values.
মানচিত্র এবং ল্যাম্বদা ফাংশন ব্যবহার করে দুটি তালিকা ভাগ করার জন্য একটি ফাংশন লিখুন।	Write a function to divide two lists element-wise using map and lambda functions.
প্রদত্ত স্ট্রিং-এর সমস্ত স্পেসকে অক্ষর * তালিকা আইটেম * তালিকা আইটেম * তালিকা আইটেম * তালিকা আইটেম '%20' দিয়ে প্রতিস্থাপনের জন্য একটি ফাংশন লিখুন।	Write a function to replace all spaces in a given string with '%20'.
একটি মিশ্র তালিকা থেকে এমনকি সংখ্যা খুঁজে পেতে একটি পাইথন ফাংশন লিখুন।	Write a Python function to extract numbers from a mixed list.

Table 3: Examples of Bangla instructions and their English translations.

caused by mistranslations or deeper reasoning gaps were rarely resolved, underscoring the limits of inference-time self-correction. The prompt used for this task is provided in Appendix A.

6.5 Translation Error Analysis

Although the LLM-based translation pipeline preserved semantics in most cases, some instructions suffered from semantic drift. A representative example is shown below:

> **Original Bangla:** "একটি চতুর্ভুজ সমীকরণের মূলগুলি একে অপরের প্রতিদ্বন্দ্বিতা কিনা তা পরীক্ষা করার জন্য একটি পাইথন ফাংশন লিখুন"
> **LLM Translation:** Write a Python function to check whether the roots of two quadratic equations are in competition with each other.

Model	Pass@1 (English)
ReasonFlux-Coder-14B	0.73
phi-4	0.74
Llama-3.1-8B	0.59
codegemma-7b	0.42
Qwen3-14B	0.75
Qwen2.5-Coder-14B	**0.81**

Table 4: Few-shot performance on translated English instructions.

Method	Pass@1
QLoRA	0.90
QLoRA + Feedback mechanism	**0.94**

Table 5: Impact of feedback-guided inference.

Here, "প্রতিদ্বন্দ্বিতা" (literally competition) was mistranslated as in competition, while the intended mathematical meaning was reciprocal. This caused the generated code to implement incorrect logic. Such cases highlight that while test-case-aware prompting improves translation, bridging natural Bangla phrasing with precise programming semantics remains challenging.

6.6 Shared Task Outcome

Our final system `Qwen2.5-Coder-14B` with QLoRA fine-tuning and feedback-guided inference achieved Pass@1 score of 0.934 on the blind evaluation set, ranking second overall in the shared task leaderboard. This demonstrates that with careful translation, parameter-efficient fine-tuning, and inference-time self-correction, open-weight LLMs can achieve state-of-the-art performance on Bangla-to-Python code generation.

7 Conclusion

This work presents an LLM-based system for generating Bangla-to-Python code. The suggested approach integrates an LLM-based translation pipeline, parameter-efficient fine-tuning with QLoRA, and a feedback-guided inference loop. These components enabled the model to achieve Pass@1 accuracies of 0.94 on the development set and 0.934 on the blind test set. These results demonstrate that, with careful translation, efficient adaptation, and test-case-aware inference, LLMs can match the performance of much larger or closed-source systems. The future aim is to utilize fine-tuned, larger-parameter models, such as those employing LoRA and other advanced techniques, to capture more nuanced, task-specific representations. We also aim to evaluate closed-source LLMs available through APIs that benefit from stronger hardware and training pipelines. Improving translation fidelity, particularly for idiomatic Bengali expressions, remains a key challenge and a crucial step toward more robust multilingual code generation.

8 Limitations

The system achieved competitive results, but several limitations persist. Methodological challenges such as translation fidelity and error-driven code correction, along with resource constraints in training and deployment, remain. Addressing these issues is essential to enhancing the robustness and scalability of Bangla-to-Python code generation systems. Key limitations include:

- Hardware constraints restricted us to QLoRA fine-tuning on a single GPU, which prevented exploration of full-precision LoRA or larger models that could offer additional improvements.
- Although generally effective, the translation pipeline occasionally introduced semantic drift, particularly with idiomatic Bangla expressions and loanwords. These errors affected code generation and were only partially addressed by the feedback mechanism.
- The feedback-guided inference loop depended on the quality of error traces. When tracebacks did not identify issues, retries seldom led to meaningful corrections.
- The experiments were limited to open-weight LLMs, which excluded evaluation of potentially stronger closed-source models that benefit from larger-scale pretraining and inference resources.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Haau-Sing Xiaocheng Li, Mohsen Mesgar, André FT Martins, and Iryna Gurevych. 2023. Python code generation by asking clarification questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14287–14306.

Shahriar Kabir Nahin, Rabindra Nath Nandi, Sagor Sarker, Quazi Sarwar Muhtaseem, Md Kowsher, Apu Chandraw Shill, Md Ibrahim, Mehadi Hasan Menon, Tareq Al Muntasir, and Firoj Alam. 2025. TituLLMs: A family of Bangla LLMs with comprehensive benchmarking. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24922–24940, Vienna, Austria. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Nishat Raihan and Marcos Zampieri. 2025. TigerLLM - a family of Bangla large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 887–896, Vienna, Austria. Association for Computational Linguistics.

CodeGemma Team, Heri Zhao, Jeffrey Hui, Joshua Howland, Nam Nguyen, Siqi Zuo, Andrea Hu, Christopher A Choquette-Choo, Jingyue Shen, Joe Kelley, and 1 others. 2024. Codegemma: Open code models based on gemma. *arXiv preprint arXiv:2406.11409*.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. 2025. Co-evolving llm coder and unit tester via reinforcement learning. *arXiv e-prints*, pages arXiv–2506.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Li Zhong, Zilong Wang, and Jingbo Shang. 2024. Debug like a human: A large language model debugger via verifying runtime execution step by step. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 851–870.

A Prompts

This appendix contains the prompts employed in system development and testing, offering additional insight into our framework.

A.1 Translation Prompt

```
messages = [
    {
        "role": "system",
        "content": """You are a professional
    translator specializing in technical and
    programming content...

Translation Guidelines:
- Preserve Technical Accuracy: Maintain exact
    meaning of programming concepts
- Keep Code Elements Intact: Preserve English
    technical terms in standard form
- Maintain Instructional Clarity: Ensure natural
     English coding instructions
- Precision Over Literal Translation: Focus on
    exact intended meaning

Example:
Input: একটি স্ট্রিং-এ একটি আক্ষরিক স্ট্রিং অনুসন্ধান করার জন্য
    একটি ফাংশন লিখুন এবং রেজেক্স ব্যবহার করে মূল স্ট্রিং-এর
    মধ্যে অবস্থানটি খুঁজে বের করুন যেখানে প্যাটার্নটি ঘটে।
Output: Write a function to search for a literal
     string within a main string and find the
    position within the main string where the
    pattern occurs using regex."""
    },
    {
        "role": "user",
        "content": f"Translate this Bangla
    coding instruction to English: {
    bangla_instruction}"
    }
]
```

A.2 Few-shot Prompt (Bangla Instructions)

Listing 1: Few-shot prompt with translated English examples

~~~
system_message = """You are an expert Python
    programmer. Your task is to generate clean,
    efficient, and correct Python functions that
     pass all given test cases.

CRITICAL RULES:
1. ALWAYS wrap your code in ```python ``` blocks
2. Write ONLY the function implementation, no
    extra explanations
3. Use the EXACT function name from the example
4. Ensure the function passes ALL test cases
5. Handle edge cases and invalid inputs
    appropriately
6. Use appropriate data types based on test case
     patterns

Here are examples of how to solve different
    types of problems:

EXAMPLE 1 - String Processing:
Task: একটি প্রদত্ত স্ট্রিং-এ প্রথম পুনরাবৃত্ত অক্ষর খুঁজে পেতে একটি
    পাইথন ফাংশন লিখুন।
Test Cases:
assert first_repeated_char("abcabc") == "a"
assert first_repeated_char("abc") == "None"
assert first_repeated_char("123123") == "1"

Expected Solution:
```python
def first_repeated_char(s):
    seen = set()
    for char in s:
        if char in seen:
            return char
        seen.add(char)
    return "None"
```
EXAMPLE 2 - Mathematical Function:
Task: প্রদত্ত পূর্ণসংখ্যাটি একটি মৌলিক সংখ্যা কিনা তা পরীক্ষা
    করার জন্য একটি ফাংশন লিখুন।
Test Cases:
assert prime_num(13) == True
assert prime_num(7) == True
assert prime_num(-1010) == False

Expected Solution:
```python
~~~

~~~
def prime_num(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n**0.5) + 1, 2):
        if n % i == 0:
            return False
    return True
```
Code Quality Standards:

- Write code with proper indentation
- Optimize for correctness first, then efficiency
- Handle common edge cases (empty inputs, None values, negative numbers, etc.)

Return the exact data type shown in test cases"""

user_prompt = f"""Generate a Python function for this problem:

Task: {instruction}

Test Cases:
{test_list}

Expected Function Name: {function_name}

Requirements:

- Follow the examples shown in the system message
- Analyze the test cases carefully to understand input/output patterns
- Implement the function to pass ALL test cases exactly
- Return the appropriate data type as shown in test cases
- Handle edge cases gracefully (empty inputs, invalid values, etc.)
- Use efficient algorithms where applicable

Generate ONLY the Python function wrapped in python blocks. No explanations needed."""
~~~

A.3 Few-shot Prompt (Translated Instructions)

Listing 2: Few-shot prompt with translated English examples

~~~
system_message = """You are an expert Python programmer. Your task is to generate clean, efficient, and correct Python functions that pass all given test cases.

CRITICAL RULES:
1. ALWAYS wrap your code in ```python ``` blocks
2. Write ONLY the function implementation, no extra explanations
3. Use the EXACT function name from the example
4. Ensure the function passes ALL test cases
5. Handle edge cases and invalid inputs appropriately
6. Use appropriate data types based on test case patterns

Here are examples of how to solve different types of problems:

EXAMPLE 1 - String Processing:
Task: Write a Python function to find the first repeated character in a given string.
Test Cases:
assert first_repeated_char("abcabc") == "a"
assert first_repeated_char("abc") == "None"
assert first_repeated_char("123123") == "1"

Expected Solution:
```python
def first_repeated_char(s):
    seen = set()
    for char in s:
        if char in seen:
            return char
        seen.add(char)
    return "None"
```
EXAMPLE 2 - Mathematical Function:
Task: Write a function to check if a given integer is a prime number.
Test Cases:
assert prime_num(13) == True
assert prime_num(7) == True
assert prime_num(-1010) == False

Expected Solution:
```python
def prime_num(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n**0.5) + 1, 2):
        if n % i == 0:
            return False
    return True
```
Code Quality Standards:

- Write code with proper indentation
- Optimize for correctness first, then efficiency
- Handle common edge cases (empty inputs, None values, negative numbers, etc.)

Return the exact data type shown in test cases"""

user_prompt = f"""Generate a Python function for this problem:

Task: {instruction}

Test Cases:
{test_list}

Expected Function Name: {function_name}

Requirements:

- Follow the examples shown in the system
~~~

```
    message
- Analyze the test cases carefully to understand
     input/output patterns
- Implement the function to pass ALL test cases
    exactly
- Return the appropriate data type as shown in
    test cases
- Handle edge cases gracefully (empty inputs,
    invalid values, etc.)
- Use efficient algorithms where applicable

Generate ONLY the Python function wrapped in
    python blocks. No explanations needed."""
```

A.4 Feedback/Retry Prompt

Listing 3: Few-shot prompt with translated English examples

```
attempt_analysis = f"""

# PATTERN ANALYSIS FROM {len(previous_attempts)}
     ATTEMPTS:
- Attempt 1: {len(previous_attempts[0])}
    characters
- Latest: {len(previous_attempts[-1])}
    characters
- Different approaches tried: {len(set(attempt
    [:50] for attempt in previous_attempts))}

# AVOID REPEATING: The same logic pattern has
    failed multiple times. Try a fundamentally
    different approach."""

    # Enhanced error analysis based on error
    type
    specific_guidance = ""
    if failed_test['status'] == '
    ASSERTION_FAILED':
        specific_guidance = """
## ASSERTION FAILURE GUIDANCE:
- Check return data type (int, str, list, tuple,
     bool)
- Verify exact return format matches expected
    output
- Consider sorting if order doesn't matter
- Handle empty cases explicitly"""
    elif failed_test['status'] == 'RUNTIME_ERROR
    ':
        error_msg = failed_test['error'].lower()
        if 'index' in error_msg or 'list' in
    error_msg:
            specific_guidance = """
## INDEX/LIST ERROR GUIDANCE:
- Check for empty list/string handling
- Verify array bounds (0 to len-1)
- Handle edge case when input is empty"""
        elif 'key' in error_msg or 'dict' in
    error_msg:
            specific_guidance = """
## DICTIONARY ERROR GUIDANCE:
- Check if key exists before accessing
- Use .get() method with default values
- Initialize dictionaries properly"""
        elif 'attribute' in error_msg:
            specific_guidance = """
## ATTRIBUTE ERROR GUIDANCE:
- Check object types before method calls
- Verify variable is initialized
- Import necessary modules"""

    # Create comprehensive error feedback
    feedback = f"""

## PREVIOUS ATTEMPT FAILED - ADVANCED DEBUGGING:

- Error Type: {failed_test['status']}
- Error Message: {failed_test['error']}
- Failing Test Case: {failed_test['test_case']}
- Failed at Test #{failed_test['index']} out of
    {len([r for r in test_results if 'index' in
    r])}

{specific_guidance}

{attempt_analysis}

# SYSTEMATIC DEBUGGING APPROACH:
1. ANALYZE INPUT/OUTPUT: What data types and
    patterns do test cases show?
2. EDGE CASE CHECK: Empty inputs, single
    elements, boundary values
3. ALGORITHM CHOICE: Is this DP, greedy, two-
    pointer, sliding window, etc.?
4. IMPLEMENTATION: Step through the failing test
     case manually
5. IMPORTS: Add math, re, collections, itertools
     if needed

# Original Task: {instruction}

# CRITICAL SUCCESS FACTORS:
- Function signature must match test case
    exactly
- Return type must match expected output
    precisely
- Handle ALL edge cases shown in test patterns
- Use efficient algorithm for the problem type

GENERATE A COMPLETELY NEW APPROACH - Previous
    attempts failed for a reason."""
```

NSU_PiedPiper at BLP-2025 Task 2: A Chain-of-Thought with Iterative Debugging Approach for Code Generation with Bangla Instruction

Ahmad Fahmid[*] [1], **Fahim Foysal**[*] [1], **Wasif Haider**[*] [1], **Shafin Rahman**[1], **Md Adnan Arefeen**[1]

[1]North South University, Dhaka, Bangladesh

Correspondence: {ahmad.fahmid,fahim.foysal02,wasif.haider}@northsouth.edu

Abstract

Code generation from natural language instructions in Bangla is a fundamental task in programming automation, as explored in BLP-2025 Shared Task 2: Code Generation in Bangla. Current code generation models are designed primarily for high-resource languages such as English, which creates major limitations when applied to Bangla. The key challenges are limited training data and difficulty in correctly interpreting Bangla programming instructions. In this paper, to accommodate Bangla instructions, we present a chain of thought (CoT) based prompting approach with Qwen2.5-Coder-14B model. We further introduce few-shot example in the prompt template to improve the accuracy. During competition, an accuracy of 93% is achieved in the shared test set (test_v1.csv) and 82.6% is achieved on the public and private test sets (hidden). After the competition is closed, we implement a debugger prompt technique which refines answers with 3 iterative fixing attempts. Applying this new technique on the public shared test set, our system outperforms by 7% and achieves 100% accuracy on the public test set, highlighting the effectiveness of combining CoT prompting with iterative debugging.

1 Introduction

With the rapid advancement of Large Language Models (LLMs) and their success in automated code generation (Roziere et al., 2024), there has been increasing interest in extending these capabilities beyond English to serve the whole world. Code generation from natural language instructions (Li et al., 2025; Yang et al., 2025) has become a fundamental task in programming automation which enables developers to express algorithmic thoughts in human language and receive executable code solutions (Roziere et al., 2024). However, despite significant progress in English-based code generation systems, the development of programming tools for low-resource languages is significantly low (Luo et al., 2024).

Bangla, spoken by over 300 million people globally and serving as the official language of Bangladesh and the second most spoken language in India (Eberhard et al., 2024), represents a critical case study for multilingual code generation. The semantic complexity of Bangla, including its unique script system and syntactic structures that differ from English, makes it challenging for existing code generation models that are mainly trained on English datasets (Raihan and Zampieri, 2025a; Wang et al., 2024). Also, since there are not many parallel Bangla-code datasets and Bangla is hardly present in programming-related data, the problem becomes even bigger (Luo et al., 2024).

Current code generation models, while achieving impressive performance on English benchmarks (Roziere et al., 2024), show excessive performance degradation when directly applied to non-English instructions like Bangla (Cassano et al., 2023). This limitation stems from several factors: (1) insufficient multilingual understanding capabilities; (2) insufficient training on multilingual programming datasets; and (3) the lack of specialized techniques to handle the semantic and syntactic structures of low-resource languages in programming contexts. Moreover, existing approaches typically treat code generation as a direct translation task without considering the multi-step reasoning often required for complex algorithmic problems expressed in natural language.

To address these challenges and bridge the gap in multilingual code generation, we present a comprehensive approach for generating Python code from Bangla natural language instructions. Our method leverages the Qwen2.5-Coder-14B-Instruct model (Team et al., 2024) enhanced with Chain-of-Thought reasoning (Zhou et al., 2024) and incor-

[*] Equal contribution.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 595–600
December 23, 2025 ©2025 Association for Computational Linguistics

porates a novel iterative debugging system specifically designed to handle the unique challenges of cross-lingual code generation (Chen et al., 2023; Liu et al., 2024). Unlike previous approaches that rely solely on direct generation, our system implements a multi-stage process that includes careful prompt engineering with Bangla instruction processing, automatic test case generation, and systematic error correction through iterative refinement.

The main contributions of our work can be summarized as follows:

1. We introduce a chain-of-thought strategy (CoT) for multilingual code generation that helps the model reason step by step across languages.

2. Furthermore, we combine it with an iterative debugging system that can automatically detect and fix errors up to three times, making the generated code from Bangla instructions more accurate and reliable.

3. During the competition, our system scored 93% accuracy on the shared public test set (test_v1.csv) and 82.6% on the combined public and private hidden sets. After the competition ended, we experimented with a simple debugger-style prompt that lets the model review and fix its own answers up to three times. With this added refinement step, our performance on the public test set improved by 7%, allowing the system to reach 100% accuracy on the publicly shared test set. This shows the significance of our step-by-step approach with debugging, and it also gives some clear ideas about the problems and solutions in multilingual code generation.

2 Related Work

In our search, we found several code generation models capable of outputting precise and usable code. The efforts of (Yin and Neubig, 2017) provided a foundational approach by using an Abstract Syntax Tree (AST) to capture the syntax of a specific programming language, enabling the generation of grammatically correct code from natural language input. This surpassed the performance of sequence-to-sequence models which often made syntactical mistakes. Building on this, more recent work by (Yin et al., 2023) addresses the challenges of modern, interactive programming environments like data science notebooks. They introduced the ARCADE benchmark and the large-scale PACHINCO model, demonstrating the importance of understanding rich, multi-turn context to generate functionally correct and relevant code for complex data analysis tasks. We have also seen efforts by (Bhattacharjee et al., 2023) to tackle the lack of unified resources in Bangla by creating BanglaNLG, which creates a comprehensive benchmark for evaluating various natural language generation tasks, and also pre-trained BanglaT5, which is a strong baseline model for the Bangla language. Addressing the performance issues of previous Bangla models, (Raihan and Zampieri, 2025b) created the TigerLLM family, where they used the NCTB textbooks and the Bangla-Instruct dataset to pretrain and finetune their model respectively. Our work improves upon these by incorporating Program-of-Thought and Chain-of-Thought reasoning as well as an iterative debugging system to output precise Python code from a Bangla Natural Language input.

3 Method

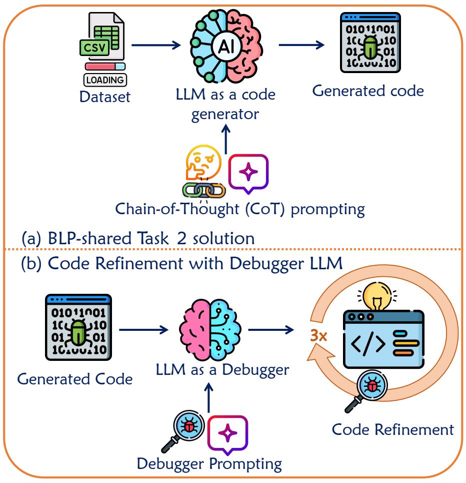
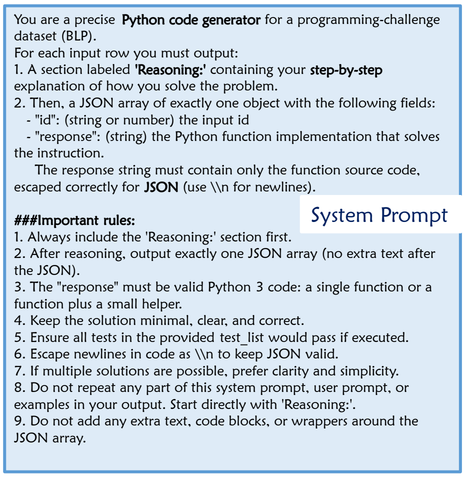


Figure 1: The proposed framework. (a) The initial CoT-based prompt technique we consider during initial submission. (b) A debugging prompt technique on the generated code is applied with maximum 3 trials to refine the code.

In this section, we discuss the Chain-of-Thought (CoT) prompt driven approach with iterative debugging.

Figure 1 refers to the two proposed approaches. (a) We first employ Chain-of-Thought prompting with the Qwen2.5-Coder-14B-Instruct model to generate Python code directly from Bangla natural-language instructions. The system and user prompt templates are shown in Figure 2 and Figure 3.

(b) To further enhance correctness, we apply a code refinement stage, where the same model is guided by a specialized debugging prompt to iteratively fix errors up to n times until all test cases pass. We consider the value of n as 3 in all our experiments. Figure 4 demonstrates the prompt template. In this, we extract function signatures with regular expressions and extend test coverage by prompting the model to generate three additional test cases. Using few-shot reasoning examples, the Qwen2.5-Coder-14B-Instruct model produces code accompanied by an explicit reasoning trace. The output JSON is parsed to recover executable Python functions for validation against the complete test suite.


System Prompt

You are a precise **Python code generator** for a programming-challenge dataset (BLP).
For each input row you must output:
1. A section labeled **'Reasoning:'** containing your **step-by-step** explanation of how you solve the problem.
2. Then, a JSON array of exactly one object with the following fields:
  - "id": (string or number) the input id
  - "response": (string) the Python function implementation that solves the instruction.
    The response string must contain only the function source code, escaped correctly for **JSON** (use \\n for newlines).

###Important rules:
1. Always include the 'Reasoning:' section first.
2. After reasoning, output exactly one JSON array (no extra text after the JSON).
3. The "response" must be valid Python 3 code: a single function or a function plus a small helper.
4. Keep the solution minimal, clear, and correct.
5. Ensure all tests in the provided test_list would pass if executed.
6. Escape newlines in code as \\n to keep JSON valid.
7. If multiple solutions are possible, prefer clarity and simplicity.
8. Do not repeat any part of this system prompt, user prompt, or examples in your output. Start directly with 'Reasoning:'.
9. Do not add any extra text, code blocks, or wrappers around the JSON array.

Figure 2: System prompt for python code generation from bangla instruction using Chain-of-Thought.

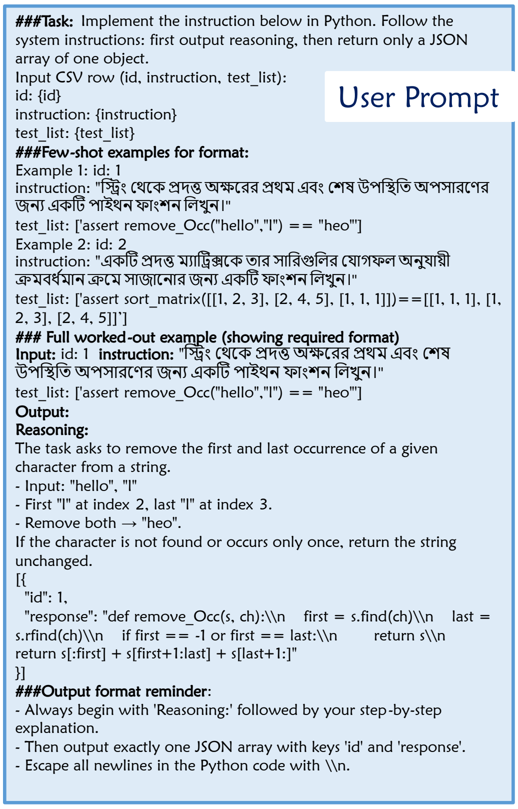
We can summarize the proposed iterative debugging approach as follows.

- Generation: The model uses the base Generation Prompt to produce initial Python code.
- Debugging: If validation fails, the previous code and error messages are fed into the Debugging Prompt to generate a corrected version. After each attempt, the JSON response is extracted, the code is run against the combined tests, and the loop stops early if all tests pass.

4 Results

Dataset

The BLP-2025 Task 2 dataset comprises three splits i.e. trial.csv, dev_v2.csv, test_v1.csv with a total

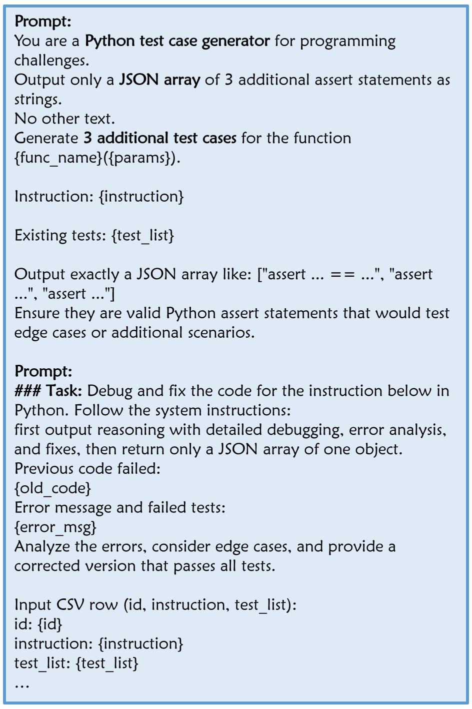

User Prompt

###Task: Implement the instruction below in Python. Follow the system instructions: first output reasoning, then return only a JSON array of one object.
Input CSV row (id, instruction, test_list):
id: {id}
instruction: {instruction}
test_list: {test_list}
###Few-shot examples for format:
Example 1: id: 1
instruction: "স্ট্রিং থেকে প্রদত্ত অক্ষরের প্রথম এবং শেষ উপস্থিতি অপসারণের জন্য একটি পাইথন ফাংশন লিখুন।"
test_list: ['assert remove_Occ("hello","l") == "heo"']
Example 2: id: 2
instruction: "একটি প্রদত্ত ম্যাট্রিক্সকে তার সারিগুলির যোগফল অনুযায়ী ক্রমবর্ধমান ক্রমে সাজানোর জন্য একটি ফাংশন লিখুন।"
test_list: ['assert sort_matrix([[1, 2, 3], [2, 4, 5], [1, 1, 1]])==[[1, 1, 1], [1, 2, 3], [2, 4, 5]]’]
### Full worked-out example (showing required format)
Input: id: 1 **instruction:** "স্ট্রিং থেকে প্রদত্ত অক্ষরের প্রথম এবং শেষ উপস্থিতি অপসারণের জন্য একটি পাইথন ফাংশন লিখুন।"
test_list: ['assert remove_Occ("hello","l") == "heo"']
Output:
Reasoning:
The task asks to remove the first and last occurrence of a given character from a string.
- Input: "hello", "l"
- First "l" at index 2, last "l" at index 3.
- Remove both → "heo".
If the character is not found or occurs only once, return the string unchanged.

```
[{
  "id": 1,
  "response": "def remove_Occ(s, ch):\\n    first = s.find(ch)\\n    last = s.rfind(ch)\\n    if first == -1 or first == last:\\n        return s\\n    return s[:first] + s[first+1:last] + s[last+1:]"
}]
```

###Output format reminder:
- Always begin with 'Reasoning:' followed by your step-by-step explanation.
- Then output exactly one JSON array with keys 'id' and 'response'.
- Escape all newlines in the Python code with \\n.

Figure 3: User prompt for python code generation from bangla instruction using Chain-of-Thought.



Prompt:
You are a **Python test case generator** for programming challenges.
Output only a **JSON array** of 3 additional assert statements as strings.
No other text.
Generate **3 additional test cases** for the function {func_name}({params}).

Instruction: {instruction}

Existing tests: {test_list}

Output exactly a JSON array like: ["assert ... == ...", "assert ...", "assert ..."]
Ensure they are valid Python assert statements that would test edge cases or additional scenarios.

Prompt:
### Task: Debug and fix the code for the instruction below in Python. Follow the system instructions:
first output reasoning with detailed debugging, error analysis, and fixes, then return only a JSON array of one object.
Previous code failed:
{old_code}
Error message and failed tests:
{error_msg}
Analyze the errors, consider edge cases, and provide a corrected version that passes all tests.

Input CSV row (id, instruction, test_list):
id: {id}
instruction: {instruction}
test_list: {test_list}
...

Figure 4: A snippet of iterative generation and debugging prompt approach.

of 974 programming problems. Each example includes an id, a Bangla natural language description-based instruction of the required function and test cases that include a list of input and output pairs for automatic evaluation. But only in trial we get another row called response which has the solution of the problem. Problems cover a diverse range of tasks, including string operations, numerical computations, data structure manipulation, and algorithmic challenges. Instruction lengths range from approximately 25 to 50 Bangla words, and task difficulty varies from simple to complex multi-step problems. A sample of dataset is shown in Table 1.

id	instruction	test_list

1	স্ট্রিং থেকে প্রদত্ত অক্ষরের প্রথম এবং শেষ উপস্থিতি অপসারণের জন্য একটি পাইথন ফাংশন লিখুন।
	Example:
	def remove_Occ(s,ch):
	  # your code
	  return s
		['assert remove_Occ(\"hello\",\"l\") == \"heo\"']

2	একটি প্রদত্ত ম্যাট্রিক্সকে তার সারিগুলির যোগফল অনুযায়ী ক্রমবর্ধমান ক্রমে সাজানোর জন্য একটি ফাংশন লিখুন।
	Example:
	def sort_matrix(M):
	  # your code
	  return M
		['assert sort_matrix([[1, 2, 3], [2, 4, 5], [1, 1, 1]])==[[1, 1, 1], [1, 2, 3], [2, 4, 5]]']

3	একটি ত্রিভুজাকার প্রিজমের আয়তন খুঁজে বের করার জন্য একটি পাইথন ফাংশন লিখুন।
	Example:
	def find_Volume(l,b,h) :
	  # your code
	  return l
		['assert find_Volume(10,8,6) == 240']

Table 1: Sample rows of test_v1.csv test set

For each CSV row with an id, instruction, and original test assertions, we first parse the existing tests. Then, we use a test-generation prompt to create three additional edge-case tests. The original and generated tests are combined to improve coverage and catch tricky scenarios during validation and debugging.

Implementation Details

We consider Qwen2.5-Coder-14B as our primary model and the debugger model. For comparison, we also consider Qwen2.5-7B model. The maximum new token generation is set to 512, temperature 0.15, top-p is set to 1. We set the maximum number of tries with the debugger to 3. We use NVIDIA RTX 5070 GPU to run the experiments locally.

Experimental Results and Performance Metrics

Our method improved code generation a lot by using chain-of-thought (CoT) prompting combined with iterative debugging. We tested this on both test_v1.csv public test set and test_v1.csv with additional private test set. During the challenge, we used test_v1.csv with additional private test set to create CoT-only solutions. As shown in Table 2, for the Qwen2.5-Coder-14B-Instruct model, we got 82.6% accuracy on the test_v1.csv with additional private test set and 93% on the test_v1.csv public test set. The smaller Qwen2.5-7B model reached 81% accuracy on the test_v1.csv public test set.

After the challenge, we improved the system with CoT + Debugger. Since we no longer had access to test_v1.csv with additional private test set, we evaluated only on test_v1.csv shared test set. With this enhanced method, Qwen2.5-7B improved to 88% accuracy (7% higher) and Qwen2.5-Coder-14B-Instruct reached a perfect score of 100% (also 7% improvement).

Model	Accuracy (CoT) (%)		Accuracy (CoT + Debug) (%)
	(test_v1.csv) + private test set	test_v1.csv	test_v1.csv
Qwen2.5-7B	—	81	**88**
Qwen2.5-Coder-14B-Instruct	82.6	93	**100**

Table 2: Accuracy comparison table using CoT and CoT+Debugger approach.

5 Ablation Study

The biggest improvement was on test_v1.csv shared test set. Debugging fixed runtime errors and handled tricky edge cases, which made the larger model perfect on test_v1.csv shared test set evaluations. This shows that our multi-stage approach is very effective at improving code quality and making sure the code passes tests.

Evaluation and Error Analysis with GPT-5: We save the raw outputs from every attempt to disk, and partial results are stored along the way so we can resume if something interrupts the process. When the pipeline finishes, we create two JSON files: one with full explanations and one with only the ID and response fields for submission. This logging helps us track each iteration, see what types of errors occurred, which errors were fixed, and which ones persisted. It allows us to understand how failures were resolved over time in the debugging workflow.

To measure the impact of debugging, error analysis is administered using GPT-5 model as a judge

Category	Question	Response (CoT)	Response (CoT + Debugger)

Correctly generated w/ only CoT	**instruction:** একটি প্রদত্ত ম্যাট্রিক্সকে তার সারিগুলির যোগফল অনুযায়ী ক্রমবর্ধমান ক্রমে সাজানোর জন্য একটি ফাংশন লিখুন।
	Example:
	def sort_matrix(M):
	    # your code
	    return M
	test_list: ["assert sort_matrix([[1, 2, 3], [2, 4, 5], [1, 1, 1]])==[[1, 1, 1], [1, 2, 3], [2, 4, 5]]"]
		✓
		{
		  "id": 2,
		  "response": "def sort_matrix(M):\n # Calculate the sum of each row\n row_sums = [sum(row) for row in M]\n\n # Sort the rows based on their sums\n sorted_rows = sorted(M, key=lambda row: sum(row))\n\n # Return the sorted matrix\n return sorted_rows"
		}
			N/A

Correctly generated w/ CoT + Debugger	**instruction:** একটি প্রদত্ত তালিকা থেকে অদ্ভুত সংখ্যা অপসারণের জন্য একটি পাইথন ফাংশন লিখুন।
	Example:
	def remove_odd(l):
	    # your code
	    return l
	test_list: ["assert remove_odd([1,2,3]) == [2]"]
		✗
		def remove_odd(l):
		    result = []
		    for num in l:
		        if num % 2 == 0:
		            result.append(num)
		    return result
			✓
			{
			  "id": 402,
			  "response": "def remove_odd(l):\n result = []\n for num in l:\n if num % 2 == 0:\n result.append(num)\n return result"}

Unable to generate correctly	**instruction:** একটি জটিল সংখ্যার কোণ পেতে একটি ফাংশন লিখুন।
	Example:
	def angle_complex(a,b):
	    # your code
	    return a
	test_list: ["assert angle_complex(0,1j)==1.5707963267948966 ", "assert angle_complex(0,1j)==1.5707963267948966 "]
		✗
		{
		  "id": 114,
		  "response": "import cmath\n\ndef angle_complex(a, b):\n return cmath.phase(complex(a, b))\n\nassert angle_complex(0, 1j) == 1.57j7963267948966\nassert angle_complex(we, 1j) == 1.57j7163267948966"}
			✗
			{
			  "id": 114,
			  "response": "import cmath\n\ndef angle_complex(a, b):\n return cmath.phase(complex(a, b))\n\nassert angle_complex(0, 1j) == 1.57j7963267948966\nassert angle_complex(we, 1j) == 1.57j7163267948966"}

Figure 5: Examples of solved problems by CoT and CoT+Debugger. An example of unsolved problems are also shown.

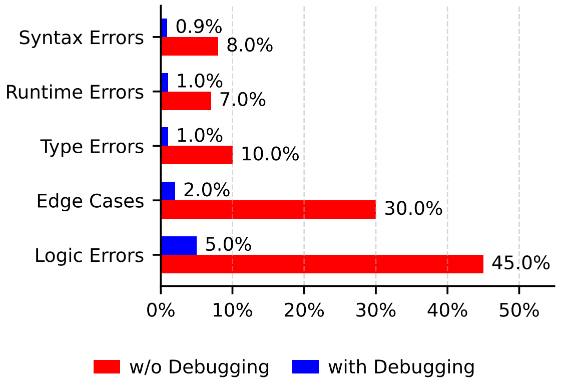


Figure 6: Using GPT-5 as a judge to analyze and track error reduction

that reveals a sharp drop in failure rates after debugging as shown in Figure 6. It shows the logic errors fall from 45% to 5%, edge-case issues from 30% to 2%, type errors from 10% to 1%, runtime errors from 7% to 1%, and syntax errors from 8% to 0.9%. These results clearly show that our debugging approach significantly improved the overall quality of the output.

6 Qualitative Analysis

We demonstrate a qualitative analysis of our approach. Examples of problems successfully solved using CoT and CoT + Debugger are shown in Figure 5. The first example is solved by CoT. In the second one, our approach correctly indents the code, hence return the correct python code. We further show a case that remains unsolved in the thrid example.

7 Conclusion

This paper addresses Bangla code generation from natural language instructions. By employing Chain-of-Thought prompting and iterative debugging, we achieved 100% on the test_v1.csv test set through systematic error correction. Future research directions include fine-tuning models on Bangla-specific programming datasets and developing larger instruction datasets tailored to Bangla programming.

References

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.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Arjun Guha, and 1 others. 2023. Multi-lingual code generation and evaluation with mbxp. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 1064–1075.

Xinyun Chen, Lin Chan, Shiyu Yu, Pengcheng Bai, Shih-wei Chen, Ed Choi, Mingsheng Chen, and Denny Zhou. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46554.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2024. *Ethnologue: Languages of the World*, 27 edition. SIL International, Dallas, Texas.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E Gonzalez, and Ion Stoica. 2025. S*: Test time scaling for code generation. *arXiv preprint arXiv:2502.14382*.

M. Liu and 1 others. 2024. Iterative debugging for neural code generation. *ACM Transactions on Programming Languages and Systems*, 46(3).

Ziyin Luo and 1 others. 2024. Multilingual code generation: Challenges and opportunities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Nishat Raihan and Marcos Zampieri. 2025a. Tiger-LLM - a family of Bangla large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Nishat Raihan and Marcos Zampieri. 2025b. Tiger-LLM - a family of Bangla large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 887–896, Vienna, Austria. Association for Computational Linguistics.

Baptiste Roziere and 1 others. 2024. Code llama: Open foundation models for code. *arXiv preprint arXiv:2408.09187*.

Qwen Team, Jinze Bai, Shuai Xu, Yankai Zhang, Zhenru Wang, Songyang Wang, Ziyan Li, Wang Wang, Li Wang, Siyuan Liu, Siyu Wang, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Y. Wang and 1 others. 2024. Recent advances in bangla natural language processing. In *Proceedings of the 2024 Asia-Pacific Language Processing Conference*.

Jian Yang, Wei Zhang, Jiaxi Yang, Yibo Miao, Shanghaoran Quan, Zhenhe Wu, Qiyao Peng, Liqun Yang, Tianyu Liu, Zeyu Cui, and 1 others. 2025. Multi-agent collaboration for multilingual code instruction tuning. *arXiv preprint arXiv:2502.07487*.

Pengcheng Yin, Wen-Ding Li, Kefan Xiao, Abhishek Rao, Yeming Wen, Kensen Shi, Joshua Howland, Paige Bailey, Michele Catasta, Henryk Michalewski, Oleksandr Polozov, and Charles Sutton. 2023. Natural language to code generation in interactive data science notebooks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 126–173, Toronto, Canada. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Denny Zhou and 1 others. 2024. Advancing chain-of-thought reasoning in large language models. *Nature Machine Intelligence*, 6:45–58.

NALA_MAINZ at BLP-2025 Task 2: A Multi-agent Approach for Bangla Instruction to Python Code Generation

Hossain Shaikh Saadi[1] **Faria Alam**[2] **Mario Sanz-Guerrero**[1]
Minh Duc Bui[1] **Manuel Mager**[1] **Katharina von der Wense**[1,3]
[1]Johannes Gutenberg University Mainz, Germany
[2]Saarland University, Germany [3]University of Colorado Boulder, USA
hsaadi@uni-mainz.de

Abstract

This paper presents the winning system for the BLP-2025 Shared Task on Code Generation from Bangla Instructions, which consists of a multi-agent pipeline. First, a code-generation agent produces an initial solution from the input instruction. The candidate program is then executed against the provided unit tests (pytest-style, assert-based). Only the failing cases are forwarded to a debugger agent, which reruns the tests, extracts error traces, and, conditioning on the error messages, the current program, and the relevant test cases, generates a revised solution. Using this approach, our submission achieves first place in the shared task with a $Pass@1$ score of 95.4. We make our code publicly available.[1]

1 Introduction

In recent years, the quality of automatically generated code based on natural language instructions has increased rapidly, and a plethora of work exists in this domain (Yang et al., 2025; Hui et al., 2024; Li et al., 2023; Huynh and Lin, 2025). However, most benchmarks and systems remain vastly English-centric (Jiang et al., 2024). This imbalance narrows access to program-synthesis tools for non-English speakers and limits our understanding of how linguistic factors – such as morphology, script variation, and code-mixing – impact the path from instructions to executable programs. Additionally, it makes it harder for developers with limited English proficiency to use these tools (Wang et al., 2023). Bangla – spoken by over 270 million people worldwide – is an example of an inadequately supported language in this area. Although NLP resources are expanding (Raihan et al., 2025a,b), there are still few high-quality evaluations and systems for Bangla instruction-to-code generation (Raihan et al., 2025b).

This paper presents the winning system for the BLP-2025 Code Generation Shared Task (Raihan et al., 2025c). Our system adopts a targeted two-agent pipeline. First, a code-generation agent produces an initial Python solution from the Bangla instruction. We then execute the generated code against a set of pytest-style unit tests from the provided dataset and an external dataset (Austin et al., 2021) to obtain concrete failure signals. Rather than re-prompting every data sample, only the failing samples together with the error traces, the current code, and the corresponding unit tests are passed to a debugger agent that proposes minimal, correctness-oriented edits. By localizing debugging to the right spots, we concentrate our inference effort and avoid needless code changes. We evaluate multiple proprietary APIs for this multi-agent approach. Empirically, this approach delivers strong results on the shared task data.

Our contributions are two-fold: (i) a simple but effective agent architecture that couples generation with test-driven refinement; (ii) a selective feedback mechanism that exposes the debugger only to failing cases and distilled execution traces.On the official leaderboard, our system achieves the best performance among all participating teams (Pass@1: 95.4%). We complement the results with a small ablation study and analysis. An overview of our proposed system is provided in Figure 1.

2 Related Work

Code generation with large language models (LLMs) is a well-established area of study (Jiang et al., 2024; Dong et al., 2025; Qi et al., 2021), with interest and capability surging in recent years (Yang et al., 2025; Hui et al., 2024; Li et al., 2023). Despite this momentum, there remains relatively little work on Bangla instruction-to-code generation (Raihan et al., 2025b). Test-driven development (TDD) is a well-established methodology in

[1] https://github.com/shaikhsaadi999/blp25_code_genneration

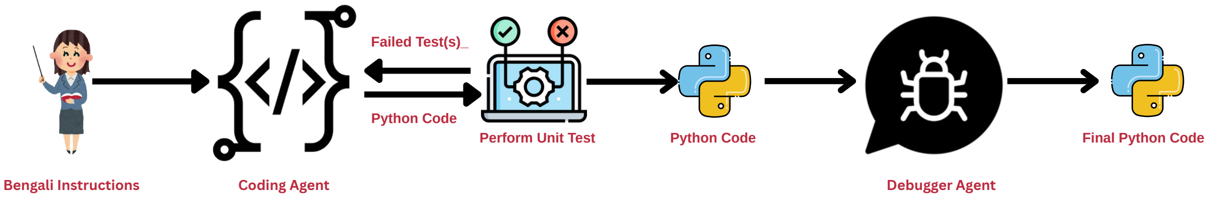


Figure 1: Multi-agent Bangla→Python code generation pipeline with selective debugging via unit test feedback.

which developers write tests before implementing the functional code, ensuring that the resulting solution directly aligns with the initial problem statements (Mathews and Nagappan, 2024). In TDD, test cases are the central artifact, and there is a growing body of research on automated test-case generation (Takerngsaksiri et al., 2024; He et al., 2025b; Wang et al., 2025; He et al., 2025a). For evaluating code generation systems, unit test-based evaluation has become the de facto standard (Chen et al., 2021). Rather than relying on text-only metrics used in machine translation (e.g., BLEU), running generated programs against hidden tests directly measures functional correctness (Huynh and Lin, 2025).

3 Task Definition

The goal of the shared task is to generate Python code from a given Bangla instruction. The instruction itself contains the given Bangla instruction, function name, and argument names. Furthermore, assert-based pytest-style unit tests (Raihan et al., 2025c) are provided for each instruction. For each instance i, we are given a Bangla natural-language instruction $x_i \in X$ which includes a function name and its argument names, and a `pytest` test suite T_i. No type signature is provided for the arguments of the given function; required behavior is determined solely by the instructions x_i and test suites T_i. A candidate program must define the function precisely as provided by the function name since `pytest` is designed with the same function names. For example, if a function's name is `min_cost`, for this function, one given assert-based unit test will be like this, `assertmin_cost([[1,2,3],[4,8,2],[1,5,3]],2,2)==8`. A program *passes* if all tests in T_i succeed, and the goal is to generate a program $\hat{p}_i$ for an instruction x_i that passes all tests. For this task, the evaluation metric is $Pass@1$ (Chen et al., 2021). This is a standard metric for code generation tasks that measures the proportion of instructions for which a single generated code passes all the provided unit tests. So, in short, given one attempt, how often does the model produce an entirely correct solution. In the most common setting, for a given instruction, one solution per problem is generated, and then the fraction of the generated solutions that pass all the unit tests is reported as a $Pass@1$ score.

4 Data

Development dataset Our development set consists of 400 Bangla instructions paired with function names, each accompanied by three unit tests. The organizers provide this dataset (Raihan et al., 2025a).

Test dataset The test set contains 500 Bangla instructions with function names, each accompanied by a single unit test, so that we gain some understanding of the behavior of the input and output of the function we need to generate. Final submitted codes are tested on hidden unit tests for the sake of fair and robust evaluation.

External dataset The task permits the use of external models and datasets. To augment the available tests for code generation, we search for a resource with matching function formats and existing unit tests. We identify Austin et al. (2021), which directly covers unit tests for 480 relevant samples. In this external dataset, the number of unit tests per instance varies: some instances are provided with three tests, while others are provided with four. For each instruction in the provided data, we match its function name against those in Austin et al. (2021), retrieve the corresponding 480 functions, and extract all associated unit tests. Any tests that do not overlap with the original test set are appended to the single provided unit test.

5 Pipeline Overview

We employ two agents: a **code generation agent** $G_\theta : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{P}$ that takes an instruction $x_i \in \mathcal{X}$ together with a test suite $T_i \in \mathcal{T}$ and returns a generated program. The provided test suite is not included in the prompt for the code generation agent; it is only used for testing. Function names and argument names are also provided inside the instruction. A **debugger agent** $D_\phi : \mathcal{X} \times \mathcal{T} \times \mathcal{P} \rightarrow \mathcal{P}'$ then takes an instruction $x_i \in \mathcal{X}$, a test suite $T_i \in \mathcal{T}$, and a buggy program $p \in \mathcal{P}$ and returns an improved program.

Stage 1 (Code generation agent) For each Bangla instruction x_i, this agent generates a Python code p and executes each unit test from the provided `pytest` test suite T_i. If it fails for any of the tests, it generates again by mentioning in the prompt that it has failed. After this second step, the code generation agent saves the generated code for each instruction x_i. If a generated code fails even after the second attempt, it is saved as *failed*. Otherwise, it is saved as *passed*.

Stage 2 (Debugger agent) In this stage, the debugger agent processes only those codes that fail in Stage 1. We call this set of failed codes $\mathcal{F}$. For each failed code $p \in \mathcal{F}$, we re-execute the provided test suites to consolidate the error trace. After that, the debugger conditions on its corresponding instruction x_p, test suite T_p, and the error traces $\widetilde{\mathcal{T}}(p)$ for the test suite to produce repaired code p', which is saved as the final generated code. Our detailed algorithm is provided in Algorithm 1.

Test case generation Apart from these two agents, we employ another agent for test case generation from the given Bangla instruction x_i, and a `pytest` test suite T_i. For each instruction, we generate five extra test cases. We employ an abstract syntax tree-based syntax evaluation technique, which guarantees that the selected unit tests are syntactically correct. This agent is not part of our primary system.

Proprietary APIs For our proposed pipeline, we use proprietary APIs from OpenAI, Google, and Anthropic. From Google, we use `Gemini-2.5-Flash`, from OpenAI, we use `GPT-5` and `GPT-4.1`, and from Anthropic, `Claude-Sonnet-4`. For our primary submission, we use `GPT-5` since it is the best-performing model in the dev set. For the code generation agent, we set the reasoning effort to `low` and for the debugger agent, we set it to `high`.

All detailed prompts for our agents are provided in the Appendix A.

Model	Pass@1	Error rate
GPT-5	**64.60**	**35.40**
Gemini-2.5 Flash	52.60	47.40
Claude sonnet 4	58.20	41.80
GPT-4.1	58.00	42.00

Table 1: Stage-1 (code generation only; Pass@1 on Test Set). Error rate is 100 − Pass@1.

Algorithm 1 Multi-agent Code Generation

```
Require: $\mathcal{X}, \mathcal{T}, G_\theta, D_\phi$
Ensure: Final code $p'$
function RunTests($p, T$)
    Run pytest suite $T$ on $p$ (exact name/args).
    return (PASS,$\varnothing$) if all pass; else (FAIL, aggregated trace $\widetilde{\mathcal{T}}(p)$).
function end
for $i = 1$ to $N$ do
▷ Stage 1: Code Generation (max 2 attempts)
    $x_i \leftarrow$ Instruction for instance $i$
    $T_i \leftarrow$ Test suite for instance $i$
    $p \leftarrow G_\theta(x_i)$;
    $(r, \_) \leftarrow$ RunTests($p, T_i$)
    if $r$ = FAIL then
        $p \leftarrow G_\theta(x_i)$;
        $(r, \_) \leftarrow$ RunTests($p, T_i$)
    SAVE($p$)
for each $p \in \mathcal{F}$ do    ▷ Stage 2: Debugger on failures only, here $\mathcal{F}$ is the set of all failed codes from the Stage 1
    $x_p \leftarrow$ Corresponding Instruction for $p$
    $T_p \leftarrow$ Corresponding Test suite for $p$
    $(r, \widetilde{\mathcal{T}}(p)) \leftarrow$ RunTests($p, T_p$)
    if $r$ = FAIL then
        $p' \leftarrow D_\phi(x_p, T_p, \widetilde{\mathcal{T}}(p))$
    SAVE($p'$)
```

6 Results & Discussion

We separately report our scores for Stage 1 (coding agent) and Stage 2 (debugger agent) in Table 1 and Table 2, respectively. We find that without the unit tests, any prompt we give to the model fails to provide the correct code almost 40% of the time. Only `GPT-5` can go above 60% in $Pass@1$ score; even

Model	Pass@1	Error rate
GPT-5	**95.4**	**4.6**
Gemini-2.5 Flash	59.80	40.20
Claude sonnet 4	79.00	21.00
GPT-4.1	82.60	17.40

Table 2: Stage-2 (debugging with error traces and unit tests; Pass@1 on Test Set). Error rate is 100 − Pass@1.

CLAUDE SONNET 4 can not go over 60%. When the test cases are presented to the model within the prompt during the debugging process in the second stage, the model's performance improves significantly, except for `Gemini-2.5-Flash`. For this model, the $Pass@1$ score improves only 13.68%, which is very low compared to the other three models. `GPT-5` attains the top Stage-1 $Pass@1$ of **64.6**, outperforming `GPT-4.1` by 6.6 points, `Claude Sonnet 4` by 6.4, and `Gemini-2.5 Pro` by 12.0. Full Stage-1 results are shown in Table 1.

In stage two, we run the test cases first, then provide the debugger with the corresponding code and its related instructions, which fail at one of the unit tests. We give the error trace and all the unit tests. We observe that when unit tests are provided, the increase in $Pass@1$ scores is significantly higher than when the code is generated without the test cases in the prompt. We observe that for GPT-5, the increment is 47.67% for `Gemini-2.5-Flash`, `Calude Sonnet 4`, `GPT-4.1`, 13.68%, 35.73%, and 41.37%, respectively. So the biggest gain is from `GPT-5`. In stage 1, we only provide the Bengali instruction and the function name, which is not enough context for the model. That's why, when we add the unit test, the model's performance improves significantly. From this, it's clear that more information about the problem context helps the model by providing the structure it needs to reason, cover edge cases, and self-debug.

Overfitting to the unit tests The scores indicate the effectiveness of incorporating the two-stage approach and augmenting test cases within the prompt. Also, there is a possibility that the models are being overfit to the provided test cases. During the development phase, we are provided with all the test cases that are used during the system's evaluation in Codabench.[2] Our two-stage system achieve a 99.8 $Pass@1$ score during the evaluation of the development phase. We use all the provided unit tests in the prompt, which are used during the evaluation. However, during the test phase, we observe that when we submit our system for evaluation, there is a significant error rate compared to the development phase due to the presence of hidden unit tests. Using the available unit tests, we can achieve a 99.2 $Pass@1$ score using GPT-5 and the evaluation script provided by the organizers locally; however, when we submit our system to Codabench to run on hidden unit tests, we obtain 95.4, which indicates that the generated codes sometimes can not handle edge cases, or they are not generalizing well, and are over-fitted to the provided unit tests.

Effect of external data In the provided test set, there is only one unit test for each instruction. We collect external test cases for 480 instructions out of 500 instructions, which we describe in Section 4.[3] We observe that without the external unit test for `GPT-5`, our proposed pipeline achieves an 86.00 $Pass@1$ score, which is significantly lower than the 95.4 $Pass@1$ score achieved by augmenting unit tests from the external dataset. This indicates the effectiveness of the more unit tests, which helps the model to generate more generalized code.

Effect of generated unit tests As we mentioned in Section 5, we also try generating test cases from the single provided test cases. We use the provided Bengali instruction, and a single test case to generate five test cases using `GPT-5` with a high reasoning effort. Out of 500 instructions, the model was able to generate at least 3 test cases for 461 instructions. Then we create the code in the same process. First, we generate without the test cases. Then we debug the code using the original and the augmented generated test cases. With the artificial test cases, we achieve an 84.00 $Pass@1$ score. Which is a lot higher than the stage 1 score (64.60) but lower than the stage 2 score (95.4) for `GPT-5`.

Effect of translation We translate the Bengali instructions into English using the GOOGLETRANS[4] library, replacing the original Bengali instructions with the translated ones. Then we follow the same two-stage pipeline. In Figure 2, we report $Pass@1$ for four models (GPT-5, GPT-4.1, Claude-Sonnet-4, and Gemini-2.5-Flash) across two evaluation stages, with and without translation. Overall, GPT-5 achieves the highest $Pass@1$ in all

[2] https://www.codabench.org/

[3] We used this external dataset after consulting with the organizers.

[4] https://pypi.org/project/googletrans/

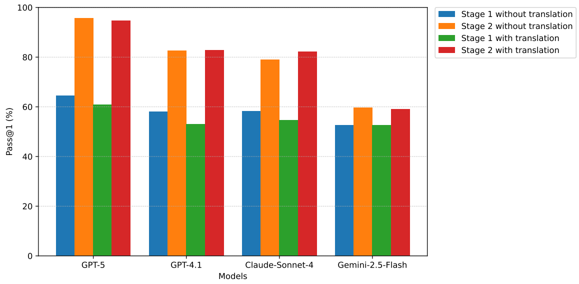


Figure 2: Pass@1 of different models across both stages, with and without translation.

conditions, followed by GPT-4.1, while Claude-Sonnet-4 and Gemini-2.5-Flash lag behind. For `Gemini-2.5-Flash`, we observe no change in either stage. In contrast, for `GPT-4.1`, the $Pass@1$ score for the Stage-1 coding agent decreases from 58.00 to 53.00, while for Stage-2, it increases from 82.60 to 82.80. We see the same phenomenon for `Claude-Sonnet-4` as well. For Stage 1 $Pass@1$ decrease from 58.20 to 54.60 and for Stage 2 it increased from 79.00 to 82.20. However, for `GPT-5` in both stages, the translation decreases performance. The decrease is much higher for stage 1. In Stage 1 the $Pass@1$ decreases from 64.60 to 60.80, and for Stage 2, it decreases from 95.80 to 94.80. These results suggest that during translation, some task-specific information is lost, which in turn hinders the Stage-1 coding agent—but can also clarify particular instructions. When test cases are presented alongside the English instructions in Stage 2, this additional signal slightly improves scores despite the decline in Stage 1. Overall, different models react differently to translation, with varying sensitivity.

7 Conclusion

This paper presents our system for the BLP-25 Shared Task on Code Generation from Bangla Instructions: a straightforward and effective multi-agent pipeline. It couples an initial coder with a selective debugger driven by unit tests and their error traces. Based on GPT-5, our system achieves a $Pass@1$ score of 95.4% and secures first place for the BLP-2025 Task 2. Our study highlights how structured feedback (error traces + tests) consistently boosts model performance over using only the instruction. Overall, the system demonstrates that targeted, test-driven refinement substantially improves code synthesis in an underserved language.

Acknowledgement

This work was supported by the Carl Zeiss Foundation through the TOPML and MAINCE projects (grant numbers P2021-02-014 and P2022-08-009I). We thank the anonymous reviewers for their feedback and suggestions.

Limitations

Our study focuses exclusively on proprietary models for this task and does not evaluate any open-source models, which limits the reproducibility and generalizability of our findings. In addition, to enriching the number of unit tests supporting our test case-driven solution, we rely on an external dataset. Without this enrichment, models achieve approximately 10 percentage points lower scores than the reported ones. Finally, we observe substantial differences in performance across the proprietary models; however, we lack transparency into their training data and procedures, making it difficult to attribute these differences to specific factors. We utilize AI assistants, specifically ChatGPT (GPT-5), to assist with editing sentences in our paper writing.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. 2025. A survey on code generation with llm-based agents. *Preprint*, arXiv:2508.00083.

Lehan He, Zeren Chen, Zhe Zhang, Jing Shao, Xiang Gao, and Lu Sheng. 2025a. Use property-based testing to bridge llm code generation and validation. *Preprint*, arXiv:2506.18315.

Zhongmou He, Yee Man Choi, Kexun Zhang, Jiabao Ji, Junting Zhou, Dejia Xu, Ivan Bercovich, Aidan Zhang, and Lei Li. 2025b. Hardtests: Synthesizing high-quality test cases for llm coding. *Preprint*, arXiv:2505.24098.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Nam Huynh and Beiyu Lin. 2025. Large language models for code generation: A comprehensive survey of challenges, techniques, evaluation, and applications. *Preprint*, arXiv:2503.01245.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *Preprint*, arXiv:2406.00515.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, and 48 others. 2023. Starcoder: may the source be with you! *Preprint*, arXiv:2305.06161.

Noble Saji Mathews and Meiyappan Nagappan. 2024. Test-driven development and llm-based code generation. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, page 1583–1594. ACM.

Weizhen Qi, Yeyun Gong, Yu Yan, Can Xu, Bolun Yao, Bartuer Zhou, Biao Cheng, Daxin Jiang, Jiusheng Chen, Ruofei Zhang, Houqiang Li, and Nan Duan. 2021. ProphetNet-X: Large-scale pre-training models for English, Chinese, multi-lingual, dialog, and code generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 232–239, Online. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *Preprint*, arXiv:2509.09101.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Wannita Takerngsaksiri, Rujikorn Charakorn, Chakkrit Tantithamthavorn, and Yuan-Fang Li. 2024. Pytester: Deep reinforcement learning for text-to-testcase generation. *Preprint*, arXiv:2401.07576.

Wenhan Wang, Chenyuan Yang, Zhijie Wang, Yuheng Huang, Zhaoyang Chu, Da Song, Lingming Zhang, An Ran Chen, and Lei Ma. 2025. Testeval: Benchmarking large language models for test case generation. *Preprint*, arXiv:2406.04531.

Zhiruo Wang, Grace Cuenca, Shuyan Zhou, Frank F. Xu, and Graham Neubig. 2023. Mconala: A benchmark for code generation from multiple natural languages. *Preprint*, arXiv:2203.08388.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

A Prompts

Coding Agent User Prompt

```
You are a strict Python coding assistant.
Write ONLY a valid Python implementation for
 the function below. Do not include tests,
imports that are not used, main guards,
comments, or explanations. Only provide the
function definition and helper code inside
the same file as needed.

{status}

Constraints:
Use the exact function name and signature.
Prefer clear, deterministic algorithms. No
randomization.
Avoid printing and input() usage.
You may define additional helper functions
and classes inside the same file if needed.
Please do not start the code with ‘‘‘python

Function signature:
def {spec.name}({’, ’.join(spec.args)}):

Also consider the problem in Bengali:
{spec.instruction_bn}

Return ONLY the code block itself, with no
backticks, quotes, or additional text.
```

Debugger Agent System Prompt

```
You are a Python debugging agent.
You receive the Instruction , current
code, failing test(s), and the error
trace.
Return ONLY corrected code. Keep
changes minimal, focused to fix
failures and consider edge cases.
Do not add comments or other non-code
text.
You may define additional helper
functions, classes and imports inside
the same file if needed.
Please do not start the code with
‘‘‘python
```

Debugger Agent User Prompt

```
Instruction (original): {instruction}
Current code: {code}
Failing tests: {failing_tests}
Error trace: {error_text}

Provide the fixed code only.
```

Unit Test Generation Agent System Prompt

```
You are a Python unit test generation
agent.
Return ONLY a Python list of strings,
each a single pytest-style assert.
Think deeply and carefully about
behavior and edge cases.
Generate constructive tests that
reveal bugs, not trivial variations.
No explanations or code fences.
Ensure asserts are syntactically
valid.
```

Unit Test Generation Agent User Prompt

```
Target function name: {func_name}
Sample assert: {sample_assert}

Generate {num_tests} new, distinct,
constructive unit tests (assert ...).
Use only Python literals in
arguments; no I/O or imports.
```

CUET_Expelliarmus at BLP2025 Task 2: Leveraging Instruction Translation and Refinement for Bangla-to-Python Code Generation with Open-Source LLMs

Md Kaf Shahrier, Suhana Binta Rashid,
Hasan Mesbaul Ali Taher, Mohammed Moshiul Hoque
Department of Computer Science and Engineering,
Chittagong University of Engineering & Technology, Chittagong 4349, Bangladesh
{u1804035,u1804037,u1804038}@student.cuet.ac.bd
moshiul_240@cuet.ac.bd

Abstract

Large language models (LLMs) have recently demonstrated strong performance in generating code from natural-language prompts. However, current benchmarks are primarily focused on English, overlooking low-resource languages like Bangla. This creates a critical research gap, as there are no well-established resources or systematic evaluations of code generation from Bangla instructions. To address the gap, we present a system that generates executable Python code from Bangla instructions. We design a two-stage pipeline: the Bangla instructions are first translated and refined into a clear English version to reduce ambiguity, and then the Python code is generated from the refined instructions through iterative error correction. For both instruction refinement and code generation, we used the open-source GPT-20B OSS model. On the official test set, our system achieves competitive results. We also analyze common errors such as unclear instructions, logical mistakes, runtime issues, and the need for external knowledge beyond the model's training data. Overall, our findings show that a simple translation–refinement pipeline can be an effective, low-cost approach to code generation in low-resource languages.

1 Introduction

Large Language Models (LLMs) have recently advanced automatic code generation, enabling systems to produce executable programs directly from natural-language instructions. Early progress was achieved with general-purpose models such as GPT-3 (Brown et al., 2020), followed by code-specific extensions such as Codex (Chen et al., 2021), CodeT5 (Wang et al., 2021), and PolyCoder (Xu et al., 2022), which significantly improved performance. Benchmarks such as HumanEval (Chen et al., 2021), APPS (Hendrycks et al., 2021), and CodeXGLUE (Lu et al., 2021) have since become standard for evaluating English-to-code generation.

However, progress in low-resource languages remains limited. Multilingual models such as XLM-RoBERTa (Conneau et al., 2020) and mBERT (Devlin et al., 2019) demonstrate cross-lingual transfer in NLP, but their application to code generation remained underexplored. In particular, Bangla, one of the most widely spoken languages globally, lacks systematic benchmarks and studies for code generation. While resources like BanglaBERT (Bhattacharjee et al., 2022) and other NLP benchmarks have supported Bangla text understanding they do not extend to executable code generation. Broader studies show that English-trained models often underperform on low-resource languages (Blasi et al., 2022). While proprietary systems (e.g., GPT-4, Claude 3) (OpenAI et al., 2023; Anthropic, 2024) and open-source initiatives (e.g., Aya, LLaMA-3) (Üstün et al., 2024; Grattafiori et al., 2024) expand multilingual coverage, systematic evaluation of Bangla instruction-to-code-generation remains absent.

To address this gap, benchmarks such as mHumanEval (Raihan et al., 2025a) and MBPP-Bangla (Raihan et al., 2025b) provide initial resources for Bangla-to-Python evaluation. Building on these, we develop a two-stage pipeline that translates and refines Bangla instructions into structured English, and then generates Python code from the refined instructions using iterative test-driven error correction. The critical contributions of the work are summarized as follows:

- We present the first systematic study on Bangla-to-Python code generation using open-source LLMs.
- We introduce a two-stage pipeline that leverages instruction translation, one-shot prompting for refinement, and zero-shot prompting for code generation.
- We incorporate a test-driven automatic cor-

rection loop to ensure the reliability of the generated code.

- We conduct comparative experiments on the official test datasets.

  All resources and code used in this study are available at: https://github.com/Hasan-Mesbaul-Ali-Taher/BLP_25_Task_2 to support reproducibility and transparency.

2 Related Work

Although several studies have been conducted on code generation in high-resource languages, this issue is at a rudimentary stage in Bengali. Chen et al. (2022) introduced CodeT, a method that uses the same pretrained LMs to auto-generate test cases and then select the best code by running samples and checking dual execution agreement and improved pass@1 across benchmarks without manual test creation. Wang and Chen (2023) reviewed LLM-based code generation and highlights both applications and evaluation methods. This also emphasizes that while LLMs significantly improved developer productivity, the assessment of generated code remains underexplored, with limited quality measures considered. Nahin et al. (2025) introduced TituLLMs, the first Bangla LLMs (1B, 3B), which are trained on 37B tokens with an extended tokenizer and evaluated on five new benchmarks. TituLLMs outperformed the initial multilingual models. Besides, they have made the TituLLMs models and benchmarking datasets publicly available. A recent study (Bhowmik et al., 2025) identified NLP challenges, evaluated 10 LLMs across eight translated datasets, found performance gaps between Bengali and English, highlighted weaknesses in smaller models, and highlighted the need for better benchmarks and datasets.

Recent works have begun to address the lack of benchmarks for Bangla-to-code generation. The *mHumanEval* benchmark (Raihan et al., 2025a) extends the HumanEval dataset to over 200 natural languages, including Bangla, and uses machine translation and expert validation for 15 languages. This resource highlights LLMs' multilingual capabilities and enables cross-lingual evaluation for code generation. Complementing this, the *Tiger-Coder* suite (Raihan et al., 2025b) introduced the first dedicated family of Bangla code LLMs (1B and 9B parameters) with a curated Bangla code instruction dataset and the MBPP-Bangla benchmark. Their models achieved 11- 18% higher Pass@1 accuracy than existing multilingual baselines and demonstrated that carefully curated data can substantially improve performance even for smaller-scale LLMs.

3 Task Description

The primary objective of this task is to automatically generate executable Python code from natural language instructions written in Bangla as described in the paper (Raihan et al., 2025c). Unlike traditional classification tasks, this problem requires models to bridge the gap between informal Bangla text and structured programming logic. A successful system must be able to (i) interpret the intent of the Bangla instruction, (ii) transform the intent into a well-defined problem specification, (iii) generate Python code that adheres to this specification, and (iv) ensure functional correctness by passing the provided test cases. This task is particularly challenging due to the scarcity of Bangla resources, the ambiguity of instructions, and the need to produce both syntactically valid and semantically correct code.

3.1 Dataset Description

The shared task organizers provided four datasets for development and evaluation at different stages. During the competition, we received the *Trial* and *Development (Dev)* datasets. After the task concluded, the *Test v1* and *Test Full* datasets were released for final benchmarking. Table 1 summarizes the key characteristics of these datasets.

Dataset	Rows	Instr.	Resp.	Tests
Trial	74	Yes	Yes	2–3
Dev	400	Yes	No	3
Test v1	500	Yes	No	1
Test Full	500	Yes	Yes	3

Table 1: Overview of datasets.

4 Methodology

The proposed approach follows a two-stage pipeline that reliably generates executable Python code from Bangla instructions. In the first stage, Bangla problem statements are translated into English and refined into well-structured specifications using a text-to-text generation model with one-shot prompting. In the second stage, these refined instructions are passed to a text-to-code generation model, which produces candidate Python solutions.

For both of the stages, we have used the open-source **GPT-20B OSS** model. To ensure functional correctness, the generated code is validated against provided test cases, and a retry mechanism is employed when failures occur. The codes are then stored in the final solution set. The overview of the entire methodology have presented in Figure 1.

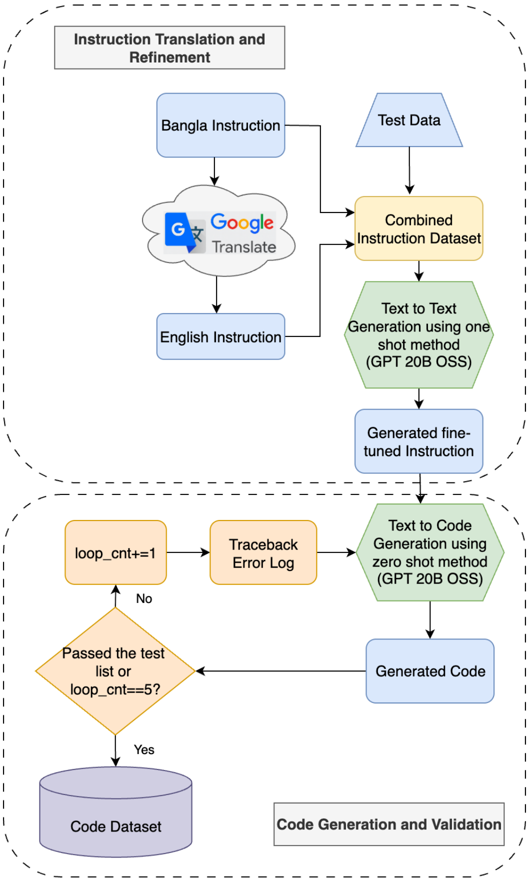

Instruction Translation and Refinement
Bangla Instruction
Test Data
Google Translate
Combined Instruction Dataset
English Instruction
Text to Text Generation using one shot method (GPT 20B OSS)
Generated fine-tuned Instruction
loop_cnt+=1
Traceback Error Log
Text to Code Generation using zero shot method (GPT 20B OSS)
No
Passed the test list or loop_cnt==5?
Generated Code
Yes
Code Dataset
Code Generation and Validation

Figure 1: Overview of the proposed two-stage pipeline for Bangla instruction to Python code generation.

4.1 Instruction Translation and Refinement

Bangla instructions b are often informal, ambiguous, and stylistically diverse, which makes direct code generation difficult. To address this, we employ a translation–refinement strategy. Each instruction b is first translated into English instruction e using Google Translate, which provides a literal but sometimes noisy version. The Bangla instruction b, its English translation e, and the test cases T are then concatenated into a single input. This input is passed to a text-to-text generation model with one-shot prompting, where an example guides the model to convert slightly noisy instructions into a refined instruction r that forms a well-structured specification. The refined output r explicitly defines the function name, parameters, return type, and task requirements. As illustrated in Algorithm 1, this refinement step ensures that the instructions fed into the code generation stage follow a consistent structure, reduce ambiguity, and support reliable code generation.

Algorithm 1 Pseudo-code of the Proposed Methodology

```
Input: Bangla instruction $b$, test list $T$
Output: Verified Python code $C$
Translate $b$ into English instruction $e$ using Google Translate.
Concatenate $\{b, e, T\}$ into a single input.
Generate refined instruction $r$ using a text-to-text model (one-shot).
Initialize loop counter $cnt \leftarrow 0$.
repeat
    Generate candidate code $c$ from $r$ using a text-to-code model.
    Execute $c$ against the test list $T$.
    if all tests pass then
        $C \leftarrow c$
        break
    else
        Update the prompt with error feedback.
    end if
    $cnt \leftarrow cnt + 1$
until $C$ is valid or $cnt = 5$
return $C$
```

4.2 Code Generation and Validation

The refined instruction r obtained from the previous stage is passed to a text-to-code generation model, which is prompted in a zero-shot setting to produce Python code c. The prompt is carefully structured to guide the model to generate only the function definition and the required logic, and to avoid unnecessary explanations or comments. This ensures that the output c remains concise and directly executable. Once a candidate solution c is generated, it is executed against the provided test list T. If the solution passes all test cases, it is accepted and stored as solution code C. If any test fails, the system captures the corresponding traceback and uses it to prompt the code generation model again. This retry mechanism is repeated up to 5 iterations, as illustrated in Algorithm 1 and Figure 1. If all test cases pass at any step, the candidate solution c is stored as the solution code C immediately. Otherwise, after five iterations, the final candidate solution c is stored regardless of outcome. This error-driven re-generation increases the likelihood of producing functionally correct solutions.

4.3 Experimental Setup

All experiments were conducted on Kaggle using a Tesla T4 GPU (16 GB VRAM), 30 GB RAM, and a dual-core CPU, running Ubuntu and Python 3.10 with preinstalled PyTorch, CUDA, and Hugging Face Transformers. This environment ensured reproducibility with minimal setup. Table 2 summarizes the hyperparameters. The prompt limit (512) sets the maximum input size, and the generation limit (1024) controls reasoning and code output length. Reasoning capacity was kept low to avoid long chains that exceeded token limits. The batch size was restricted to 2 due to GPU constraints. A low temperature (0.1) ensured deterministic outputs, and a top-p (0.9) temperature balanced diversity, avoiding unlikely tokens.

Hyperparameter	**Value**
Prompt Token Limit	512
Generated Token Limit	1024
Reasoning Capacity	Low
Batch Size	2
Temperature	0.1
Top-p	0.9

Table 2: Hyperparameters used in the Kaggle T4 GPU experiments.

5 Results and Analysis

To evaluate our system, we conducted experiments across five folds of the test set, each consisting of 500 instances. At each fold, the model generated a candidate code per instruction, and the results were normalized by dividing the number of correct codes by 500. For evaluation, codes were first validated against a single test case; if at least one candidate passed, it was then checked against all test cases. When multiple candidates passed all tests, the shortest solution (measured in character length) was selected for final validation. We adopt this criterion because shorter programs generally use fewer tokens. This choice reduces the likelihood of unnecessary complexity and of including redundant or unintended operations that do not affect the program's functional correctness. Moreover, selecting the shortest candidate discourages the model from relying on specific patterns that may satisfy the test cases by accident, without truly generalizing. This makes the evaluation more consistent and keeps the comparison focused on the essential logic rather than stylistic variations.

During the shared task evaluation phase, the proposed system initially achieved an accuracy of *0.370*. This was primarily due to indentation errors in the generated code, which caused only single-line return statements to be accepted. After addressing this issue, we observed a significant improvement in performance as shown in Table 3. The results are reported under three experimental settings.

1. **Bangla Instruction + Test Cases:** Raw Bangla instructions with provided test cases passed directly to the code generation model.
2. **Refined Instruction:** Translated and refined English instructions containing explicit function definitions, parameter types, return values, and test cases.
3. **Refined Instruction + Error Log:** The refined instruction is further augmented with error feedback from failed generations, allowing iterative re-generation.

Model	**F1**	**F2**	**F3**	**F4**	**F5**
Bangla Instr. + Test Cases	0.104	0.108	0.108	0.112	0.128
Refined Instruction	0.860	0.870	0.876	0.876	0.884
Refined Instr. + Error Log	0.860	0.904	0.910	0.920	0.924

Table 3: Accuracy across folds (F1–F5). Each fold corresponds to 500 test instances.

Table 3 reports fold-wise accuracies for the three evaluation settings. Performance is measured across five folds of 500 instances each, using the same candidate-generation and test-case–based validation procedure described earlier. It also highlights the steady improvement from one setting to the next and reports a peak accuracy of 0.924.

5.1 Ablation Study

To quantify the individual contributions of each component in our pipeline, we conduct an ablation across three variants: (1) **Raw Bangla Instruction + Test Cases**, (2) **Refined Instruction**, and (3) **Refined Instruction + Error Log**. The results are presented in Table 3. Each setting introduces exactly one additional step and allows us to isolate the impact of instruction refinement and error-driven regeneration. As shown in Figure 2, the refinement step yields the largest performance gain, taking a massive leap in the accuracy from an average of ∼0.112 to ∼0.873. This demonstrates that translating and restructuring the instruction into a precise English specification drastically reduces the ambiguity of the Bangla prompts. Adding error logs provides a further boost in the accuracy, achieving a peak accuracy of 0.924 in the fifth fold. The improvement is consistent across all folds. This demonstrates that the error feedback helps correct runtime and logical mistakes that persist even after the refinement step.

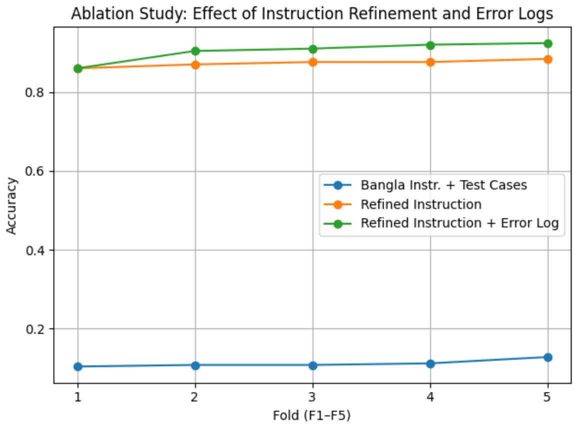


Figure 2: Ablation study comparing accuracy across folds for different pipeline variants.

Overall, the ablation indicates that *instruction refinement* is the most influential component of the pipeline, whereas the *error-driven regeneration* provides complementary gains. Together, these components make our two-stage pipeline highly effective for Bangla-to-Python code generation in low-resource settings.

6 Conclusion

This study introduced a two-stage pipeline for Bangla-to-Python code generation. The pipeline first translates and refines Bangla instructions, then generates code with test-driven correction. Experiments on shared-task datasets showed that direct Bangla-to-code generation gave suboptimal results. In contrast, the proposed refinement and error feedback loop led to substantial improvements, reaching an accuracy of up to 0.924. This work is the first systematic approach to Bangla code generation and shows that translation and refinement are effective for low-resource programming tasks. Future work may extend this approach to larger multilingual large language models and more comprehensive Bangla code benchmarks. Task-specific models could be evaluated to enhance translation, refinement, and code generation. For example, the current system uses the same GPT 20B OSS model for both instruction refinement and code generation. Future research could investigate using a specialized text-oriented model to refine instructions and another model optimized for code generation. Such specialization may yield significant performance gains. The pipeline could also be adapted to support more programming languages and complex tasks. Model selection may be broadened to include smaller yet more capable models, such as the Qwen family, which could achieve high accuracy with reduced computational resources. Advanced methodologies, such as AI agents operating sequentially to refine instructions and generate code, may also be explored to improve automation and overall system performance.

Limitations

This study comes with several shortcomings. First, the translation step relies on an external tool, i.e., Google Translate, which may introduce noise or semantic shifts in Bangla instructions and may potentially affect downstream code generation. Second, the experiments were constrained by limited computational resources and conducted in small batch sizes and restrictive hyperparameters. Third, our evaluation was conducted exclusively on the available shared-task datasets, which remain limited in scale and diversity. Moreover, the framework has only been tested with a single open-source model, GPT-20B OSS, and its generalizability to other large language models or programming languages has not yet been explored.

References

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku — model card. https://www.anthropic.com/claude-3-model-card.

Abhik Bhattacharjee, Tahmid Hasan, Kazi Samin Mubasshir, Md Saiful Islam, Wasi Uddin Ahmad, 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.

Shimanto Bhowmik, Tawsif Tashwar Dipto, Md Sazzad Islam, Sheryl Hsu, and Tahsin Reasat. 2025. Evaluating llms' multilingual capabilities for bengali: Benchmark creation and performance analysis. *Preprint*, arXiv:2507.23248.

Damián Blasi, Antonios Anastasopoulos, and Graham Neubig. 2022. Systematic inequalities in language technology performance across the world's languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5487–5510. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2022. Codet: Code generation with generated tests. *Preprint*, arXiv:2207.10397.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Andrea Grattafiori and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Ethan Arora, Collin Guo, Dawn He, Dawn Song, and Jacob Steinhardt. 2021. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Dongdong Ma, Shujie Zhou, Sining Liu, Duyu Jiang, Jiahai Lin, Daxin Tang, and 1 others. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Shahriar Kabir Nahin, Rabindra Nath Nandi, Sagor Sarker, Quazi Sarwar Muhtaseem, Md Kowsher, Apu Chandraw Shill, Md Ibrahim, Mehadi Hasan Menon, Tareq Al Muntasir, and Firoj Alam. 2025. Titullms: A family of bangla llms with comprehensive benchmarking. *Preprint*, arXiv:2502.11187.

OpenAI, Josh Achiam, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Ahmet Üstün, Viraat Aryabumi, Zheng-Xin Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. 2024. Aya model: An instruction finetuned open-access multilingual language model. *arXiv preprint arXiv:2402.07827*.

Jianxun Wang and Yixiang Chen. 2023. A review on code generation with llms: Application and evaluation. In *2023 IEEE International Conference on Medical Artificial Intelligence (MedAI)*, pages 284–289.

Yue Wang, Weishi Wang, Shafiq Joty, Steven C.H. Lin, and Hwee Tou Ng. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8696–8708.

Frank F Xu, Suraj Nair, Shuyan Lin, Pengcheng Yin, and Graham Neubig. 2022. Polycoder: A model for programming by example in multiple languages. *arXiv preprint arXiv:2202.13169*.

AlphaBorno at BLP-2025 Task 2: Code Generation with Structured Prompts and Execution Feedback

Mohammad Ashfaq Ur Rahman[1], **Muhtasim Ibteda Shochcho**[1], **Md Fahim**[2,3]
[1]Independent University, Bangladesh
[2]Center for Computational & Data Sciences
[3]Penta Global Limited
{imashfaqfardin, sho25100, fahimcse381}@gmail.com

Abstract

This paper explores various prompting strategies in the BLP-2025 Shared Task 2, utilizing a pipeline that first translates Bangla problem descriptions into English with GPT-4o, then applies techniques like zero-shot, few-shot, chain of thought, synthetic test case integration, and a self-repair loop. We evaluated four LLMs (GPT-4o, Grok-3, Claude 3.7 Sonnet, and Qwen2.5-Coder 14B). Our findings reveal that while traditional methods like few-shot and chain-of-thought prompting provided inconsistent gains, the integration of explicit unit tests delivered a substantial performance boost across all models. The most effective strategy combined zero-shot prompting with these synthetic tests and a self-repair loop, leading GPT-4o to achieve a top Pass@1 score of 72.2%. These results represent the value of using explicit constraints and iterative feedback in code generation, offering a solid framework that improves the model's code generation capabilities.

1 Introduction

Automatic code generation is a key research area in natural language processing (NLP). This research is driven by benchmarks like HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), which evaluate how well systems can turn natural language descriptions into executable programs.

Recent large language models (LLMs) have shown strong results on English-focused benchmarks. However, applying these methods to under-resourced languages like Bangla is challenging (Kabir et al., 2023; Ahmed et al., 2025; Raihan et al., 2025b). Previous studies highlight the difficulties in building effective NLP systems for Bangla (Bhattacharjee et al., 2022; Islam et al., 2021). These challenges arise from limited resources, diverse domains, and ambiguous text forms. To bridge this gap, recent work has introduced multilingual and Bangla-specific resources for code generation. Raihan et al. (2025a) presented mHumanEval, a multilingual extension of HumanEval covering over 200 languages, including Bangla, highlighting performance gaps between high and low-resource languages. Building on this, Raihan et al. (2025b) proposed MBPP-Bangla alongside TigerCoder, a suite of Bangla-focused LLMs, reporting improvements on Bangla code generation tasks.

The Bangla Language Processing (BLP) Workshop 2025 has further advanced this area through a shared task dedicated to Bangla code generation (Raihan et al., 2025c). In this task, participants are required to create Python programs based on Bangla problem descriptions. Evaluation is conducted using a hidden set of unit tests, with systems receiving credit only if their outputs successfully pass all test cases. This setup presents several challenges: first, the problem descriptions may be linguistically ambiguous or domain-specific; second, no reference solutions are available at the time of testing; and third, models must be able to generalize to hidden tests that go beyond the visible examples. These factors make the task a rigorous benchmark for multilingual program synthesis.

To address these challenges, we explore various prompting strategies for LLMs in a shared task context. We created a pipeline that translates Bangla instructions into English with GPT-4 and uses several prompting techniques for code generation. Our study focuses on instruction-only prompting, synthetic test cases, chain-of-thought reasoning, and self-repair loops using execution feedback. We evaluate four models: Qwen2.5-Coder 14B, GPT-4o, Grok-3, and Claude 3.7 Sonnet within these frameworks. Through this comparison, we aim to illuminate how translation and prompting choices affect performance on hidden test cases, providing a practical foundation for future research in multilingual code generation.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 615–623
December 23, 2025 ©2025 Association for Computational Linguistics

2 Background

The Bangla Language Processing (BLP) Workshop 2025 introduced Task 2: Code Generation in Bangla, where the objective is to synthesize Python programs from Bangla natural language instructions (Raihan et al., 2025c). The task follows an execution-based evaluation: systems are credited only if the generated code passes all hidden unit tests. This setting requires models to generalize beyond visible examples and to handle ambiguous or underspecified problem statements, which are common in real-world competitive programming tasks.

For our experiments, we used the official development and test datasets released for the shared task. The development set contains 400 problems, while the test set includes 500 problems. Each problem consists of (i) a Bangla instruction describing the task, (ii) one public test list provided in the form of Python assertions, and (iii) hidden test cases used for leaderboard evaluation. The datasets build upon prior multilingual code generation resources, including mHumanEval (Raihan et al., 2025a) and MBPP-Bangla (Raihan et al., 2025b), which extend HumanEval-style and MBPP-style problems to Bangla.

3 Method

We developed our pipeline using a systematic, staged approach, as illustrated in Figure 1. The process starts with an initial Bangla programming instruction. GPT-4o then performs two tasks in parallel: translating the instruction into English and generating synthetic test cases for edge behaviors. The translated instruction is combined with a selected prompting strategy (e.g., Zero-shot, Few-shot, CoT, or Zero-shot with synthetic tests) and provided to a large language model (LLM) for code generation. The solution is evaluated through run tests, and if it fails, the code enters a self-repair loop for up to three attempts before producing the final code. This design allowed us to effectively assess the contributions of each component.

3.1 Translation Step

We started by running inference on native Bangla instructions using the closed-source model Qwen 2.5-Coder 14B, but performance was limited, with many outputs failing to meet expectations. This confirmed our hypothesis that the model struggled with Bangla programming instructions.

To address this, we introduced a translation step, converting problem instructions to English via GPT-4o, known for its multilingual capabilities. Testing this with Qwen 2.5-Coder 14B revealed improved accuracy: 40.4% for original Bangla instructions versus 46.4% for those translated by GPT-4o. This improvement led us to adopt translation as a standard procedure in our experiments to enhance evaluation reliability and support our structured prompting and self-repair mechanisms.

3.2 Prompting Strategies

We experimented with five prompting setups after establishing translated baselines. Each setup received translated instructions and was evaluated using Pass@1. Below, we illustrate each strategy with a representative problem from the dataset.

Zero-shot. The baseline tests how well the model can generate code from instructions without examples, using a strict format to evaluate its problem-solving ability separately from in-context learning.

Zero-shot Prompt

~~~
System prompt:
You are a Python code generator.

STRICT RULES:
- Output ONE fenced code block only:
     ```python ...```
- Inside: exactly ONE function
    definition.
- Function name + args must match
    instruction/tests.
- No helpers/classes unless required
    .
- No comments, explanations, or text
     outside code.
- Only standard library. No I/O.
- Return values only.

User prompt:
Instruction:
Write a function to calculate the
    sum of the digits of each number
    in a given list.
~~~

Few-shot. In this baseline, we provide the model with a few complete task-solution examples after giving it the actual problem statement. This technique tests the model's in-context learning ability, allowing it to infer patterns from the provided demonstrations. Find the prompt in the Appendix A.1

Chain of Thought (CoT). This method prompts the model to follow structured thinking steps before generating code. By explicitly guiding it to

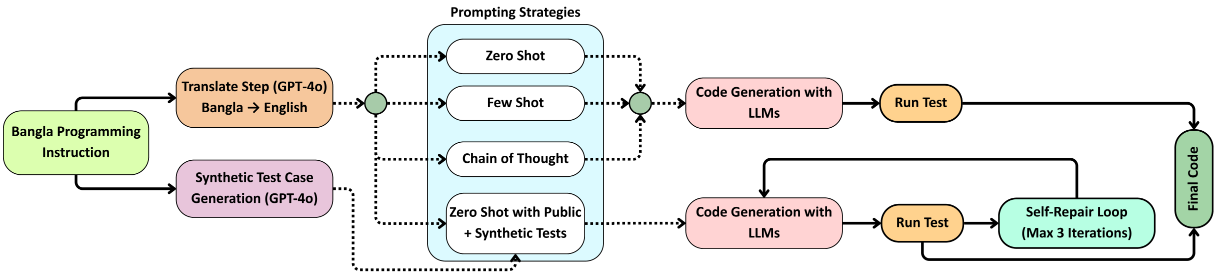

Prompting Strategies
Bangla Programming Instruction
Translate Step (GPT-4o) Bangla → English
Synthetic Test Case Generation (GPT-4o)
Zero Shot
Few Shot
Chain of Thought
Zero Shot with Public + Synthetic Tests
Code Generation with LLMs
Run Test
Code Generation with LLMs
Run Test
Self-Repair Loop (Max 3 Iterations)
Final Code

Figure 1: An overview of the complete code generation pipeline. From an initial Bangla programming instruction, GPT-4o performs English translation while also generating synthetic test cases to capture edge behaviors. The translated instruction is then combined with a selected Prompting Strategy (e.g., Zero-shot, Few-shot, CoT, or Zero-shot with synthetic tests). This combined prompt is fed to an LLM for Code Generation. The resulting code is evaluated by Run Test. If the tests fail, the code enters a Self-Repair Loop (for up to three iterations), where the model attempts to correct its errors. After that, the Final Code is produced.

consider edge cases and constraints, we encourage the model to build a more robust mental plan, leading to a more reliable final implementation. Find the prompt in the Appendix A.2

Zero-shot with public + synthetic tests. This technique provides the model with the problem instruction, with a public test case and a set of synthetic unit tests. Instead of solved examples, these tests act as explicit specifications, guiding the model by defining correct behavior and edge cases.

Zero-shot with Public + Synthetic Tests

```
System prompt:
[Same STRICT RULES as above]

User prompt:
Instruction:
Write a Python function to remove
    the first and last
occurrence of a given character from
     a string.

You must satisfy the following tests
    :

# Public test from dataset
assert remove_Occ("hello", "l") == "
    heo"

# Synthetic edge cases (robustness
    checks)
assert remove_Occ("", "x") == ""
assert remove_Occ("a", "a") == ""
assert remove_Occ("aaaa", "a") == ""
assert remove_Occ("abc", "z") == "
    abc"

Extra requirements:
- Handle empty inputs, negatives,
    duplicates, and large values.
- Preserve ordering if required.
- Ensure deterministic output.
```

```
- Match exact return types (e.g.,
    int vs float, string casing).

OUTPUT:
One fenced Python code block with
    the function only.
```

3.3 Synthetic Hidden Test Generation

To strengthen generalization to the hidden Codabench evaluator, we expanded each problem with an additional set of synthetic test cases. We generated these test cases using GPT-4o, which we prompted to propose inputs that capture edge conditions not covered by the public assertions. We instructed the model to consider empty inputs, extreme numerical values, unusual string structures, type-boundary behaviors, and other corner cases that commonly expose weaknesses in naive implementations. For each problem, GPT-4o produced eight candidate tests. To maintain consistency, we used a standard prompt template when instructing GPT-4o to augment the test set. Here is the prompt we used to generate the test cases:

Test Case Generation Prompt

```
Generate a set of synthetic Python
    assert statements based on a
    provided "function_signature" and
     a list of "public_tests".

Your goal is to create additional
    tests that cover failure-prone
    scenarios not addressed by the
    public tests, including empty/
    null inputs, boundary conditions,
     type mismatches, data-specific
    cases (e.g., unusual strings,
    large numbers, list variations),
    and any semantic corner cases.
```

~~~
CRITICAL: You must not repeat any of
    the provided "public_tests" and
    must return your response only as
     a valid JSON object containing a
     single key, "synthetic_tests",
    which holds an array of strings,
    where each string is a complete,
    runnable Python assert statement.

### INPUT:
function_signature: "def bell_Number
    (n):"
public_tests: "['assert bell_Number
    (2) == 2', 'assert bell_Number(3)
     == 5', 'assert bell_Number(4) ==
     15']"

### EXPECTED OUTPUT FORMAT:
```json
{
  "synthetic_tests": [
    "assert bell_Number(0) == 1",
    "assert bell_Number(1) == 1",
    "assert bell_Number(5) == 52",
    "assert bell_Number(6) == 203",
    "assert bell_Number(10) ==
       115975"
  ]
}
~~~

After the synthetic tests were generated, they were finalized and reused across all models: GPT-4o, Grok-3, Claude 3.7 Sonnet, and Qwen 2.5-Coder. This ensured that every system was evaluated under the same augmented constraints. In practice, these tests made the required behaviors much clearer and improved the models' robustness by providing more explicit signals regarding the expected handling of edge cases.

3.4 Self-repairing Loop

When generated code fails the provided unit tests, we initiate a self-repair loop. The model is provided with the specific AssertionError and prompted to revise its code. This feedback cycle repeats for a maximum of three attempts, allowing the model to iteratively correct its own mistakes based on the test outcomes.

Self-repairing Loop

```
Your previous attempt failed this
    assertion:
AssertionError: assert sum_of_digits
    (0) == 0
Please correct the code while
    keeping the same
function name.
```

3.5 Evaluation Protocol

We report Pass@1, defined as the percentage of problems for which the generated solution passed all asserts. Two levels of evaluation were used:

- **Public asserts** included in the dataset (to check immediate consistency).
- **Hidden tests** on the Codabench leaderboard (to assess generalization).

4 Experiments and Results

We evaluated our pipeline on four models: three closed-source (GPT-4o, Grok-3, Claude 3.7 Sonnet) and one open-source baseline (Qwen2.5-Coder 14B). Our evaluation metric is **Pass@1**, defined as the proportion of problems where the generated solution passed all hidden test cases on the Codabench leaderboard.

4.1 Results & Findings

The results, presented in Table 1, reveal the clear effect of prompt design on model performance. Our analysis highlights several key findings. First, baseline prompting strategies produced inconsistent and often suboptimal outcomes. While Claude 3.7 Sonnet achieved the highest zero-shot score (56.6%), the conventional techniques of few-shot and Chain-of-Thought (CoT) prompting did not guarantee improvements. For instance, few-shot prompting substantially degraded performance for Qwen2.5-Coder (from 46.4% to 41.2%), and CoT underperformed relative to the zero-shot baseline in three out of four models.

Second, the most remarkable performance gain came from providing explicit requirements through synthetic tests. Augmenting the zero-shot prompt with unit tests caused a substantial improvement for all models, with improvements ranging from +13.4 to +19.2 percentage points. This suggests that for code generation, defining concrete behavioral constraints is far more effective than providing abstract examples or reasoning hints.

Finally, the self-repair loop provided an additional, consistent boost to all models, pushing them to their peak performance. This iterative refinement process added a further 1.4 to 3.4 points. In the final configuration, the top-performing models converged, with GPT-4o achieving the highest score at 72.2%, followed closely by Claude 3.7 Sonnet (71.8%) and Grok-3 (71.4%).

Prompting Strategy	Hidden Tests (Pass@1 %)			
	GPT-4o	Grok-3	Claude 3.7 Sonnet	Qwen2.5-Coder
Zero-shot	54.2	52.8	56.6	46.4
Few-shot	56.8	55.4	56.2	41.2
Chain of Thought (CoT)	56.2	53.2	54.6	43.4
Zero-shot + Synthetic Tests	70.2	68.8	70.4	65.6
Zero-shot + Tests + Self-Repair	**72.2**	**71.4**	**71.8**	**69.0**

Table 1: Results comparing prompting strategies across four models. Performance generally increases with more sophisticated prompts, with the combination of explicit synthetic tests and a self-repair loop yielding the strongest results.

4.2 Failure Analysis

The evaluation revealed three main categories of failures: Code Generation Errors, Logical and Algorithmic Flaws, and Source Prompt and Localization Errors. Each category is discussed below.

Code Generation Errors

These failures occurred when the generated code did not conform to the required interface, even though the underlying logic was correct. A common case was when the model produced the right implementation but used a different function name than the one expected by the test case. For example, a function expected to be named `remove_dirty_chars` was instead generated as `str_to_list`, leading to a runtime error because the required entry point could not be found.

Logical and Algorithmic Flaws

Most failures fell into this category. The code executed without error but produced incorrect results due to incomplete algorithms, mishandling of edge cases, or inefficient approaches. As an example, a task required a function to return `-1` when no duplicate element was present in an array. The model's implementation correctly identified duplicates but returned `None` when none existed, causing the output to fail against the expected test condition. For additional failure analysis, please refer to the Appendix A.3.

4.3 Error Analysis

To understand how prompting strategies affected model performance, we analyzed common failure modes and their resolution. Three categories of errors were observed: (i) *interface alignment*, where the logic was correct but mismatched function signatures or formats caused failures; (ii) *algorithmic refinement*, where initial solutions were incomplete and improved once execution feedback clarified the requirements; and (iii) *edge-case handling*, where models overlooked inputs such as negative numbers, empty lists, or formatting constraints.

Zero-shot, few-shot, and CoT prompts often revealed these weaknesses but rarely fixed them. By contrast, adding synthetic test cases made the requirements explicit and guided the models toward outputs that matched the evaluator. Detailed case studies with code transitions are provided in Appendix A.4.

5 Conclusion

This work reveals that a multi-stage pipeline, beginning with machine translation and followed by advanced prompting, is a useful strategy for Bangla code generation. The subsequent use of synthetic tests and self-repair achieved a peak Pass@1 score of 72.2% with GPT-4o. While this approach is effective, considerable challenges remain in overcoming the models' tendency to misinterpret technically precise instructions, even when the code generation instructions themselves are accurate.

Limitations

The pipeline's effectiveness is primarily limited by its reliance on the quality of the initial Bangla-to-English translation, which can introduce errors or lose important nuances. Additionally, there is a more subtle yet noteworthy limitation that goes beyond handling prompts with inherent errors. The failure analysis shows that models often struggle with prompts that are factually correct but include technically specific terms or constraints, which can lead to misinterpretation. This highlights a key challenge in ensuring that models accurately interpret precise instructions.

References

Kawsar Ahmed, Md Osama, Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2025. Bennumeval: A benchmark to assess llms' numerical reasoning capabilities in bengali. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17782–17799.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, and Rifat Shahriyar. 2022. Banglanlg and banglat5: Benchmarks and resources for evaluating low-resource natural language generation in bangla. *arXiv preprint arXiv:2205.11081*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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.

Mohsinul Kabir, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Mir Tafseer Nayeem, M Saiful Bari, and Enamul Hoque. 2023. Benllmeval: A comprehensive evaluation into the potentials and pitfalls of large language models on bengali nlp. *arXiv preprint arXiv:2309.13173*.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

A Appendix

A.1 Few-shot Prompt.

Few-shot Prompt

```
System prompt:
[Same STRICT RULES as above]

User prompt:
Instruction:
Write a Python function to remove
    the first and last
occurrence of a given character from
     a string.

Example 1:
Instruction: Write a function to
    sort a given matrix in ascending
    order.
Solution:
def sort_matrix(matrix):
    [example implementation here]

Example 2:
Instruction: Write a function that
    counts the most common words in a
     list.
Solution:
def most_common_word(words):
    [example implementation here]
```

A.2 Chain of Thought (CoT) Prompt.

CoT Prompt

```
System prompt:
[Same STRICT RULES as above]

User Prompt:
Instruction:
{instruction}

Guidance:
- Before coding, carefully consider
    edge cases (empty input,
    negatives, duplicates, large
    values).
- Make sure return types exactly
    match expectations (int vs float,
     casing in strings, etc.).
- Then output only the final Python
    function in one fenced code block
    .

OUTPUT:
One fenced Python code block with
    exactly one function.
```

A.3 Failure Analysis

Misinterpretation and Missed Constraints

A smaller but critical set of failures was caused by the model's misinterpretation of specific problem requirements, where the prompt was accurate but the model either misunderstood a key technical term or ignored a formatting constraint. For instance, when instructed to generate a function for Woodall numbers (of the form $n \cdot 2^n - 1$), the model produced a correct implementation for the more common Mersenne numbers (of the form $2^n - 1$), effectively missing the critical n multiplier in the formula. In another case, the instruction to "set all odd bits" presented an indexing ambiguity; the model assumed 0-based indexing while the evaluator expected 1-based, leading to failure. Similarly, a task requiring explicit status strings like "Found a match!" failed when the model returned the raw search result, correctly solving the logic but ignoring the strict output format. These cases demonstrate that failures can stem from the model's misinterpretation of terminology or its tendency to overlook precise specifications.

A.4 Error Analysis

This appendix provides detailed examples of how prompting strategies influenced failure to success transitions. We organize the errors into three categories: Interface Alignment Errors, Algorithmic Refinement, and Edge-Case and Output-Type Handling. For each, we show representative cases and the corresponding code transitions, illustrating how synthetic tests corrected baseline shortcomings.

1. Interface Alignment Errors

These tasks failed in the baseline prompts because the function names or output formats did not match what the evaluator expected. Synthetic tests clarified the required interface, which led to corrections.

Example A: Removing digits from strings Zero-shot / Few-shot / CoT: The correct logic was implemented in functions named `remove_digits_from_list`, `remove_digits`, or `remove_digits_from_strings`. The evaluator, however, expected a function named `remove`, so these solutions failed. Synthetic Tests: By making the expected signature explicit, the function was renamed to `remove` while retaining the same logic.

Wrong Code

```
- def remove_digits_from_list(strings
    ):
-     return [''.join(filter(lambda c
    : not c.isdigit(), s)) for s in
    strings]
```

Repaired Code

```
+ def remove(strings):
+     return [''.join(filter(lambda c
    : not c.isdigit(), s)) for s in
    strings]
```

Example B: Converting tuples to strings Zero-shot / Few-shot: Returned comma- or space-separated strings (e.g., `', '.join(map(str, tup))`). CoT: Cast the tuple directly to a string, which preserved parentheses and commas. Synthetic Tests: Required concatenation without separators. The revised version used `”.join(tup1)`, which matched the evaluation format.

Wrong Code

```
- def tup_string(tup1):
-     return ', '.join(map(str, tup1)
    )
- def tup_string(tup1):
-     return ' '.join(map(str, tup1))
- def tup_string(tup1):
-     return str(tup1)
```

Repaired Code

```
+ def tup_string(tup1):
+     return ''.join(tup1)
```

2. Algorithmic Refinement

In these tasks, the baseline prompts produced incomplete or incorrect algorithms. Synthetic tests made the requirements explicit, guiding the model toward correct implementations.

Example A: Top-k frequent elements in nested lists Zero-shot / Few-shot / CoT: Counted only the outer list elements, failing when input contained nested lists. Synthetic Tests: The corrected version flattened the input, used `collections.Counter`, and returned the top-k elements with `heapq.nlargest`.

Wrong Code

```
- def top_k_frequent(nums, k):
-     c = {}
-     for n in nums:
-         c[n] = c.get(n, 0) + 1
-     heap = [(-freq, num) for num,
    freq in c.items()]
-     heapq.heapify(heap)
-     return [heapq.heappop(heap)[1]
    for _ in range(k)]
```

Repaired Code

```
+ def func(nums, k):
+     if k <= 0:
+         return []
+     flattened_nums = [num for
    sublist in nums for num in
    sublist]
+     frequency_counter = Counter(
    flattened_nums)
+     return heapq.nlargest(k,
    frequency_counter.keys(),
+                            key=
    frequency_counter.get)
```

Example B: Splitting a string into a list Zero-shot / Few-shot / CoT: Used `list(string)`, which split into characters. Synthetic Tests: Showed that the intended behavior was splitting into words, corrected with `string.split()`.

Wrong Code

```
- def string_to_list(string):
-     return list(string)
```

Repaired Code

```
+ def string_to_list(string):
+     return string.split()
```

Example C: Sum of amicable numbers Zero-shot: Double-counted amicable pairs. Few-shot: Simplified divisor logic but still risked duplication. CoT: Added a checked set but omitted a boundary condition. Synthetic Tests: Added both a checked set and a guard condition to avoid double counting and ensure correct limits.

Wrong Code

```
- def amicable_numbers_sum(limit):
-     total_sum = 0
-     for num in range(2, limit + 1):
-         divisor_sum =
    sum_of_divisors(num)
-         if divisor_sum != num and
    sum_of_divisors(divisor_sum) ==
    num:
-             total_sum += num
-             if divisor_sum <= limit
    :
-                 total_sum +=
    divisor_sum
-     return total_sum
```

Repaired Code

```
+ def amicable_numbers_sum(limit):
+     total_sum = 0
+     checked = set()
+     for num in range(2, limit + 1):
+         if num not in checked:
+             div_sum =
    sum_of_divisors(num)
+             if div_sum != num and
    div_sum <= limit:
+                 if sum_of_divisors(
    div_sum) == num:
+                     total_sum +=
    num + div_sum
+                     checked.add(num
    )
+                     checked.add(
    div_sum)
+     return total_sum
```

3. Edge-Case and Output-Type Handling

These tasks failed in baseline prompts because of missing sentinel values, inconsistent return types, or overlooked corner cases. Synthetic tests prompted the model to address these issues.

Example A: Decimal to binary Zero-shot / Few-shot / CoT: Returned integers instead of strings and did not handle negative inputs. Synthetic Tests: Corrected by always returning strings and explicitly handling zero and negative values.

Wrong Code

```
- def decimal_To_Binary(N):
-     if N == 0:
-         return "0"
-     binary = ""
-     while N > 0:
-         binary = str(N % 2) +
    binary
-         N = N // 2
-     return int(binary)
```

Repaired Code

```
+ def decimal_To_Binary(N):
+     if N < 0:
+         return "-" +
    decimal_To_Binary(-N)
+     elif N == 0:
```

```
+         return "0"
+     binary = ""
+     while N > 0:
+         binary = str(N % 2) +
    binary
+         N = N // 2
+     return binary
```

Example B: Maximum occurrence Zero-shot / Few-shot / CoT: Returned only the element(s) or None. Synthetic Tests: Corrected to return both the element and its frequency.

Wrong Code

```
- def max_occurrences(nums):
-     if not nums:
-         return None
-     count = Counter(nums)
-     max_count = max(count.values())
-     most_common_items = [item for
    item, freq in count.items() if
    freq == max_count]
-     return most_common_items[0] if
    len(most_common_items) == 1 else
    most_common_items
```

Repaired Code

```
+ def max_occurrences(nums):
+     from collections import Counter
+     if not nums:
+         return None, 0
+     count = Counter(nums)
+     max_item = max(count, key=count
    .get)
+     return max_item, count[max_item
    ]
```

Example C: Returning long words Zero-shot / Few-shot / CoT: Treated the input as a list of words, failing when given a raw string. Synthetic Tests: Corrected by splitting the string explicitly.

Wrong Code

```
- def long_words(n, words):
-     return [word for word in words
    if len(word) > n]
```

Repaired Code

```
+ def long_words(n, words):
+     return [word for word in words.
    split() if len(word) > n]
```

PyBhasha at BLP-2025 Task 2: Effectiveness of Semantic-Aware Translation and Ensembling in Bangla Code Generation

Foyez Ahmed Dewan[*], **Nahid Montasir Rifat**[*]
Department of Computer Science and Engineering
Rajshahi University of Engineering & Technology, Rajshahi, Bangladesh
foyez.ruet767@gmail.com, nahidmuntasir2@gmail.com

Abstract

In this paper, we present our submission to Task 2 of the BLP-2025 shared task on code generation from Bangla instructions. Our approach focused on enhancing instruction quality through translation and improving model performance with a two-stage ensemble strategy. We evaluated two proprietary and several open-source models under three instruction settings: original Bangla instructions, Bangla instructions translated into English using Facebook NLLB, and instructions rewritten in English with GPT-4.1. Experimental results showed that GPT-4.1-rewritten instructions consistently achieved the highest accuracy across models. For final predictions, we used a two-stage ensemble, achieving a *pass@1* score of **80.0%** on the hidden test set and securing 12th place on the official leaderboard. Additionally, we conducted a qualitative analysis of selected translations to illustrate how variations in instruction phrasing influenced model outputs.

1 Introduction

Code generation is the task of creating computer programs automatically from natural language descriptions. It allows users to write instructions in plain language and have a model produce the corresponding executable code. Beyond practical applications, it has emerged as a key benchmark for evaluating the reasoning and problem-solving performance of large language models (LLMs). Recent systems such as Codex, CodeLLaMA, and DeepSeek-Coder have achieved strong results on several English-centric programming benchmarks, approaching state-of-the-art performance under favorable conditions (Rozière et al., 2024; Guo et al., 2024). Yet, this success does not extend uniformly to low-resource languages like Bangla, where code generation remains underexplored due to limited training data, inconsistent instruction formats, and unreliable translation quality. Consequently, models often misinterpret Bangla prompts, producing syntactic errors or semantically incorrect code.

To address these challenges, we participate in Task 2 of the BLP-2025 shared task, which focuses on generating Python code from Bangla instructions. Our work investigates the role of instruction formulation in improving code generation quality. Specifically, we compare three input styles: raw Bangla instructions, English translations generated by Facebook NLLB, and refined English instructions rewritten by GPT-4.1. This analysis highlights how translation quality and semantic clarity directly affect model performance. While instruction reformulation substantially improves model outputs, we also explore a simple two-stage ensemble strategy to further enhance accuracy and robustness.

In summary, the main contributions of this work can be outlined as follows:

- We provide a performance comparison of proprietary, multilingual, and Bangla-centric LLMs in a one-shot code generation setting.
- We demonstrate that high-quality English reformulations of Bangla instructions substantially improve code generation outcomes.
- We introduce a simple yet effective two-stage ensemble strategy that achieves higher accuracy than standalone models.

2 Related Work

Recent Bangla-centric work such as TigerCoder (Raihan et al., 2025b) and BongLLaMA (Zehady et al., 2024) demonstrates that targeted Bangla instruction datasets and language-specific fine-tuning can improve Bangla text-to-code generation. While these studies highlight the potential of multilingual and Bangla-centric models, performance still lags

[*] Equal contribution.

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 624–628
December 23, 2025 ©2025 Association for Computational Linguistics

behind English benchmarks, leaving room for further exploration of strategies to advance Bangla code generation. Iskander et al. (2024) show that dataset quality has a major effect on LLM performance: noisy or misaligned instruction–code pairs degrade results, while filtered and curated subsets lead to significant gains even with less data, underscoring the importance of high-quality training data in low-resource scenarios. Vahtola et al. (2025) demonstrate that GPT-4 is highly effective at paraphrasing and following linguistic instructions, making it well suited for producing semantically faithful reformulations in translation-based workflows where clarity and fidelity are essential. Brown et al. (2020) further show that in-context learning improves code generation, with few-shot setups often outperforming one-shot and zero-shot; however, one-shot prompting remains widely used because it provides minimal context while ensuring consistency across models, making it suitable for controlled comparisons. Ensembling has also been explored in code generation, with systems like AlphaCode boosting pass rates through candidate reranking (Li et al., 2022). However, lightweight fallback ensembles remain largely unexplored in low-resource contexts such as Bangla. Building on these findings, our work investigates how instruction reformulation and ensemble strategies can improve Bangla code generation under a one-shot setup, addressing both the challenges of translation quality and the limitations of low-resource settings.

3 Task Description

The primary objective of this shared task (Raihan et al., 2025c) is to develop systems capable of generating Python code from Bangla programming instructions. Participating models must synthesize accurate Python solutions that satisfy the corresponding unit tests. Evaluation is performed strictly based on whether the generated code passes all test cases. By linking Bangla-language instructions to executable Python programs, this task advances research in multilingual code generation and contributes to building robust systems for low-resource languages.

3.1 Dataset Description

The dataset for this task was released in three splits: trial set, dev set, and test set. The sample distribution is summarized in Table 1. Each sample consists of a Bangla instruction describing a programming problem. The trial set additionally provides reference solutions together with the complete set of unit tests. In contrast, neither the dev set nor the test set contains reference solutions. The dev set includes the full unit test suite, while the test set offers only a single visible test case, with the remaining cases kept hidden for final evaluation.

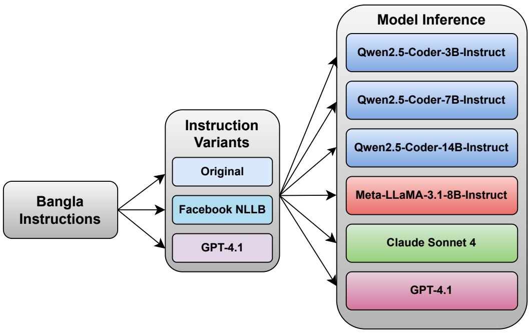


Figure 1: Model inference pipeline using three instruction styles (Original Bangla, Facebook NLLB, GPT-4.1) across multiple models.

Data Splits	Total Samples
trial	74
dev	400
test	500

Table 1: Overview of the Task 2 dataset splits.

The dev and test sets use a subset of Bangla-translated versions of the original English mHumanEval and MBPP datasets, as introduced in (Raihan et al., 2025a,b).

4 Experiments

We conducted a series of experiments to identify effective strategies for Bangla-to-Python code generation. All experiments were carried out on Kaggle using an NVIDIA P100 GPU with 16 GB memory. To enable efficient evaluation of large models (up to 14B parameters) within this limited hardware budget, we performed inference using 4-bit quantization through the `bitsandbytes` library. The subsequent subsections detail the model selection process, the instruction variants we explored, and our proposed ensemble approach.

4.1 Model Selection

We evaluated a set of open-source and proprietary LLMs on the dev set and test set under one-shot prompting, as presented in Figure 1 . One-

Model Name	Dev Phase			Test Phase		
	Original	NLLB	GPT-4.1	Original	NLLB	GPT-4.1
TigerLLM-9B-it	72.5	65.5	77.3	58.6	56.8	63.6
Qwen2.5-Coder-3B-Instruct	52.0	57.0	67.0	47.0	48.4	56.2
Qwen2.5-Coder-7B-Instruct	62.0	66.0	73.0	59.8	64.4	69.2
Qwen2.5-Coder-14B-Instruct	77.0	76.0	82.0	67.2	68.2	76.0
Meta-LLaMA-3.1-8B-Instruct	48.0	50.0	56.0	45.6	47.2	52.0
Claude Sonnet 4	–	–	82	–	–	72.4
GPT-4.1	–	–	77.8	–	–	71.0
Two-Stage Ensemble (Qwen2.5-Coder-14B-Instruct + Claude Sonnet 4)	–	–	–	–	–	**80.0**

Table 2: *pass@1* scores across three instruction variants (Original Bangla, Facebook NLLB translation, and GPT-4.1 rewriting) in both dev and test phases.

shot prompting was selected to provide minimal context while ensuring consistency across models, thereby enabling a fair comparison of their code generation capabilities. The models considered included Bangla-centric models such as `TigerLLM-9B-it`, and multilingual models including `Qwen2.5-Coder-3B/7B/14B-Instruct` (Hui et al., 2024), `Meta-Llama-3.1-8B-Instruct`, as well as proprietary models `Claude Sonnet 4` and `GPT-4.1`. This evaluation allowed us to compare the effectiveness of models specifically tailored for Bangla against those trained on broader multilingual corpora.

4.2 Instruction Variants

To investigate the effect of instruction formulation on model performance, we experimented with three variants of the input instructions during the dev phase: (i) the original Bangla instructions provided in the dataset, (ii) English translations generated using the Facebook NLLB translation model, and (iii) English instructions rewritten from Bangla using GPT-4.1. For fairness, all models were evaluated under the same experimental setup across these three variants. This design enabled us to gain a deeper understanding of how different instruction styles influenced code generation quality.

4.3 Ensemble Approach

To maximize code generation success, we employed a two-stage ensemble strategy using the best-performing models from the dev phase, as illustrated in Figure 2. `Qwen2.5-Coder-14B-Instruct` was used as the primary model to generate Python code for all instructions. Any samples that failed their unit tests were then re-generated by `Claude Sonnet 4`, enabling the secondary model to recover from errors made by the primary. This sample-level ensemble leveraged complementary strengths to increase the overall success rate.

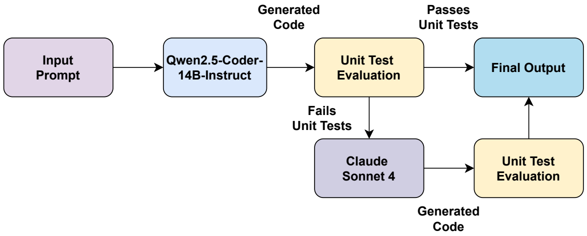


Figure 2: Two-stage ensemble strategy: The primary model (Qwen2.5-Coder-14B-Instruct) generates a code response which is evaluated via unit tests. If the code fails, the same prompt is passed to a secondary model (Claude Sonnet 4) for regeneration.

5 Results and Analysis

5.1 Initial Evaluation

We evaluated all models using the **pass rate** (*pass@1*) metric. Each model was tested in three types of instructions as described in Section 4.2

Results are reported on both dev sets and test sets, with evaluations performed against complete unit test suites. As shown in Table 2, among proprietary models, `Claude Sonnet 4` achieved the highest accuracy on both the dev and test sets when paired with GPT-4.1-translated instructions. Similarly, among the open-source models, `Qwen2.5-Coder-14B-Instruct` consistently outperformed others, even surpassing proprietary models. This highlights the strong generalization capability of Qwen when paired with high-quality instruction translations.

Other open-source models, such as `Meta-Llama-3.1-8B-Instruct` and `TigerLLM-9B-it`, also showed marked im-

id	Bangla Text	Facebook NLLB	GPT-4.1
33	প্রদত্ত অ্যারেতে সমস্ত সংখ্যার জোড়ার xor এর যোগফল খুঁজে পেতে একটি পাইথন ফাংশন লিখুন।	Write a Python function to find the sum of all the numbers in the given array.	Write a Python function to find the sum of the XOR of all pairs of numbers in a given array.
131	একটি শঙ্কুর পার্শ্বীয় পৃষ্ঠতল এলাকা খুঁজে পেতে একটি ফাংশন লিখুন।	Write a function to find a cornered side area.	Write a function to find the lateral surface area of a cone.

Figure 3: Examples of translation quality, showing how GPT-4.1 preserves semantic intent more accurately than Facebook NLLB for complex Bangla instructions.

provements when paired with GPT-4.1-based instructions, underscoring the importance of clear and semantically rich prompts in instruction-to-code generation tasks.

5.2 Ensemble Approach Outcome

We leveraged the two-stage ensemble strategy described in Section 4.3 to further boost performance. The ensemble approach achieved a **pass rate** (*pass@1*) of **80.0%** on the hidden test set under full unit test evaluation. This score represents a measurable improvement over the standalone performance of `Qwen2.5-Coder-14B-Instruct` (76.0%) and `Claude Sonnet 4` (72.4%). By selectively routing failed generations from Qwen to Claude, the ensemble effectively recovered additional correct outputs, confirming its utility in minimizing failure cases.

5.3 Impact of Instruction Formulation

To better understand why instructions translated via GPT-4.1 consistently outperformed others, we manually examined a few representative translations to assess how accurately and clearly each model conveyed the original instruction's meaning. Two such examples from the dev set are presented in Figure 3.

These examples demonstrate that GPT-4.1 translations more faithfully preserve semantic precision than those produced by Facebook NLLB, particularly for mathematically or structurally complex tasks. For instance, GPT-4.1 correctly interpreted the Bangla term "পার্শ্বীয় পৃষ্ঠতল" as "lateral surface area," whereas NLLB rendered it as "cornered side area," a phrase lacking geometric validity. Similarly, in a case involving the XOR operation, GPT-4.1 accurately preserved the intent to compute the sum of XORs over all pairs, while NLLB reduced the instruction to a basic summation task.

Bangla-native models such as `TigerLLM` achieved stronger performance when paired with either the original Bangla instructions or GPT-4.1-translated English, likely due to the increased semantic richness and clarity. In contrast, NLLB-translated prompts introduced ambiguity, leading to degraded outcomes. For all other models (e.g., `Qwen`, `LLaMA`), we observed a consistent performance ranking: GPT-4.1 > NLLB > Original Bangla, reflecting their English-dominant training distributions.

These findings suggest that, beyond simple language alignment, the semantic clarity and task specificity of instructions are critical for improving code generation quality.

6 Conclusion

In this work, we explored the impact of instruction translation on Bangla code generation tasks, with a focus on translation quality and ensemble modeling. Our evaluation demonstrated that GPT-4.1-translated instructions substantially improved performance compared to both raw Bangla and NLLB-generated instructions. We also conducted qualitative analyses to explain model sensitivities to different instruction styles. Future work will focus on fine-tuning Bangla-native models using high-quality synthetic prompts to further enhance generalization.

Limitations

At the time of our experiments, the GPT-5 API had not yet been publicly released; therefore, all instruction-rewriting experiments were conducted using GPT-4.1. While rewriting instructions in English with GPT-4.1 yielded clear improvements, our approach was limited by the absence of fine-tuning on Bangla-specific data. We expect that performance could be further enhanced through targeted fine-tuning of Bangla-centric models, as well as multilingual models trained directly on Bangla instructions. More broadly, our work highlights the lack of large, high-quality Bangla code-generation datasets; addressing this gap through dataset creation and model adaptation remains an important direction for future research.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *Preprint*, arXiv:2401.14196.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Shadi Iskander, Sofia Tolmach, Ori Shapira, Nachshon Cohen, and Zohar Karnin. 2024. Quality matters: Evaluating synthetic data for tool-using LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4958–4976, Miami, Florida, USA. Association for Computational Linguistics.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, and 7 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *Preprint*, arXiv:2509.09101.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Teemu Vahtola, Songbo Hu, Mathias Creutz, Ivan Vulić, Anna Korhonen, and Jörg Tiedemann. 2025. Analyzing the effect of linguistic instructions on paraphrase generation. In *Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025)*, pages 755–766, Tallinn, Estonia. University of Tartu Library.

Abdullah Khan Zehady, Safi Al Mamun, Naymul Islam, and Santu Karmaker. 2024. Bongllama: Llama for bangla language. *Preprint*, arXiv:2410.21200.

AdversaryAI at BLP-2025 Task 2: A Think, Refine, and Generate (TriGen) System with LoRA and Self-Refinement for Code Generation

Omar Faruqe Riyad
Shahjalal University of
Science and Technology
riyad.omf@gmail.com

Jahedul Alam Junaed
Shahjalal University of
Science and Technology
nowshadjunaed@gmail.com

Abstract

In this paper, we propose a system for generating Python code from Bangla prompts. Our approach fine-tunes open-source models with parameter-efficient techniques and leverages proprietary models via prompting. To enhance the reasoning of smaller models, we adopt a Chain-of-Thought (CoT) augmented fine-tuning, enabling them to learn intermediate reasoning steps before generating code. A self-refinement loop further improves performance by iteratively critiquing and correcting code based on execution feedback. We also employ few-shot prompting to guide inference more effectively. Applied to both open-source and proprietary models, this pipeline achieved its best results with Gemini 2.5 Pro, where our system ranked 4th on the competition leaderboard with a Pass@1 score of 0.85. We conclude with a detailed analysis of these findings.

1 Introduction

LLMs have rapidly advanced natural language processing and reasoning, achieving strong results in machine translation (Zhu et al., 2023; Feng et al., 2024), summarization (Zhang et al., 2024), dialogue (Wang et al., 2024), and complex reasoning (Lai et al., 2024). Their ability to interpret natural language and produce contextually appropriate outputs has opened new possibilities for both research and applications.

Within this broader progress, code generation has emerged as a promising direction. By translating natural language instructions into executable programs, LLMs can accelerate development, support education, and lower the barrier to programming. Benchmarks such as HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) show that state-of-the-art models can generate correct code from English prompts, underscoring their potential as programming assistants.

However, this progress is largely confined to high-resource languages, leaving languages like Bangla remain overlooked (Joel et al., 2024). The lack of datasets and tools limits training and evaluation, and direct translations often introduce errors or miss linguistic nuances. Closing this gap is vital for fairness, inclusivity, and broader access to programming. In this paper, we present a system for Bangla prompt to Python code generation. Our key contributions are:

- Adapting a pretrained LLM with LoRA for lightweight specialization.
- Expanding limited training data through a silver-to-gold augmentation strategy that generates and verifies high-quality examples.
- Enriching LLM training with two styles of chain-of-thought: concise hint-style and detailed step-by-step reasoning.
- Designing an iterative execution-feedback loop that allows the model to debug and improve its own solutions across multiple refinement steps.

2 Related Work

Large language models (LLMs) have transformed code generation, moving from rule-based systems (Gulwani, 2011) to Transformer architectures (Vaswani et al., 2017). Pretrained models like Codex (Chen et al., 2021), CodeT5 (Wang et al., 2021), and CodeGen (Nijkamp et al., 2022) set the paradigm of mapping natural language directly to executable code, establishing new program synthesis benchmarks.

Beyond full fine-tuning, parameter-efficient approaches such as LoRA (Hu et al., 2022) allow task adaptation with minimal overhead. Instruction tuning with synthetic data, including Evol-Instruct and CodeAlpaca (Luo et al., 2023; Chaudhary, 2023), further improves generalization. Reinforcement learning with human or execution feedback also aligns outputs with functional correctness (Ouyang et al., 2022; Le et al., 2022).

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 629–641
December 23, 2025 ©2025 Association for Computational Linguistics

In parallel, code-focused LLMs such as Code Llama (Roziere et al., 2023), StarCoder (Li et al., 2023), DeepSeek-Coder (Guo et al., 2024), and WizardCoder (Luo et al., 2023) show that domain adaptation significantly boosts performance over general-purpose LLMs. Yet, most advances target high-resource languages like English. Models perform far worse with low-resource languages such as Bangla (Joel et al., 2024). Translation-based methods, multilingual pretraining, and cross-lingual prompting offer partial solutions (Zhu et al., 2023; Feng et al., 2024). Recently, Raihan et al. (2025b) introduced TigerCoder, a Bangla code LLM that surpassed multilingual baselines and showed translation alone cannot close the gap, emphasizing the need for native-language resources.

3 Task and Dataset

3.1 Task Overview

The Code Generation shared task (Raihan et al., 2025c) required generating Python programs from Bangla problem prompts. A solution was correct if it passed all hidden unit tests, with evaluation performed in a sandbox environment under resource constraints. Systems were ranked by Pass@1, the percentage of prompts solved correctly.

3.2 Dataset Overview

Participants were provided with trial, dev (Raihan et al., 2025a), and test (Raihan et al., 2025b) datasets. Each entry included a Bangla instruction, a reference Python solution (trial set only), and unit tests in the form of assert statements. During training, reference solutions and public tests were available, while at evaluation, only prompts were given. Appendix B shows an example of data and datasets distribution.

4 System Description

Our submission is built upon TriGen (Think, Refine, Generate), a multi-strategy approach that leverages both parameter-efficient fine-tuning of open-source models and advanced prompt engineering of large-scale, proprietary models. We describe our two primary systems below.

4.1 System A: Fine-tuning of Open Source models

4.1.1 Base Model Selection

We began by evaluating several open-source, instruction-tuned models, including Llama3 (3B) and Qwen3 (4B), to establish a performance baseline. As shown in Table 2, the Qwen3-4B-Instruct model demonstrated superior initial performance and was selected as the base model for our later experiments. We utilized LoRA (Hu et al., 2022), to train only a small set of adapter layers, keeping the base model's weights frozen. This approach reduces computational requirements while maintaining high performance.

4.1.2 Data Augmentation and Translation

To expand our limited training data, we adopted a "silver-to-gold" data augmentation strategy (Riyad et al., 2023). We used a powerful proprietary model, Gemini 2.5 Pro, to generate high-quality solutions for the entire dev set. We then executed these generated solutions against the provided test cases and filtered for only those that passed, creating a verified dev set. This high-quality dataset was then merged with the original trial set to create an augmented training corpus (trial + verified_dev).
To investigate the impact of language alignment with our model's English pre-training, we created a pure-English version of our augmented training set using Gemini 2.5 Pro for translation.

4.1.3 Chain-of-Thought (CoT) as Hint and Step

To further improve the model's logical reasoning, we integrated CoT into our fine-tuning process. The core idea behind CoT is that prompting a model to generate intermediate reasoning steps improves its ability to solve complex problems (Wei et al., 2022; Gonzalez et al., 2024). We extend this concept to "reasoning-augmented fine-tuning" (Chung et al., 2024). The hypothesis is that by training the model on examples that explicitly include a reasoning plan (Instruction + Plan -> Code), the model internalizes the process of algorithmic decomposition. We used LLMs to generate two distinct styles of reasoning:

- **Hint-style CoT**: Strategy-level cues (e.g., "divide the number", "compute gcd", "apply multiplication"), without revealing intermediate solutions or the final output.
- **Step-by-Step (SbS) CoT**: Detailed, sequential reasoning steps that explicitly describe the path toward the solution.

We then integrated the training set with this reasoning plan. Our results showed that while the hint-style CoT, which provides only high-level cues, led to a modest improvement, fine-tuning with explicit

SbS-style CoT produced a substantial gain in accuracy.

4.1.4 Few-Shot Prompting

We designed a system prompt to enforce the competition's strict output requirements and guide the model's logic. The prompt explicitly instructs the model to follow the function name and signature, handle edge cases, produce self-contained code, and generate necessary helper classes. We utilized a few-shot approach within the system prompt by providing a concrete example of the desired response.

4.1.5 Self-Refinement Loop

We implemented a self-refinement loop (Figure 1) inspired by (Madaan et al., 2023).

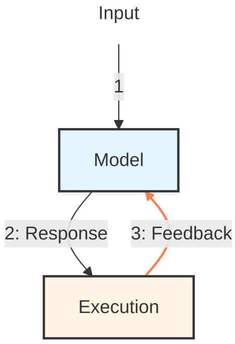


Figure 1: Execution and self-refinement loop

Think (Initial Generation): The model produces an initial code solution, which is executed against the provided public test case.

Refine (Self-Correction): For failed solutions, we construct a refinement prompt containing the original instruction, the failed code, and the corresponding error message or failed assertion from the execution environment. The model is then prompted to act as a debugger and generate a corrected solution.

Generate (Final Output): The refined code produced in the previous step becomes the new candidate solution. Iterating this loop up to three times yielded the best results.

4.2 System B: Gemini 2.5 Pro with few-shot and self-refinement

While our System-A performed well, we achieved our top-performing result by leveraging a large-scale model, Gemini 2.5 Pro. We combined few-shot prompting with self-refinement loop (Section-4.1.4 and Section-4.1.5). This approach achieved a final Pass@1 score of 0.85 on the hidden test set, outperforming all of our locally fine-tuned models.

5 Results

We began by fine-tuning several models on the trial set to establish a baseline. Among them, Qwen3-4B-Instruct achieved the best initial performance with a Pass@1 of 0.58 on the dev set (Table 2), so we used it for the later experiments.

Our first key finding was that translating the dataset to English, contrary to our initial hypothesis, resulted in a slight performance degradation to 0.50, suggesting that potential semantic shifts during translation outweighed the benefits of aligning with the model's primary pre-training language. We therefore proceeded with the Bangla-centric dataset for all subsequent fine-tuning experiments.

As shown in Table 1, each subsequent strategy yielded incremental gains. Augmenting the training data with a verified dev set improved the Pass@1 score to 0.54. Integrating Chain-of-Thought (CoT) as high-level hints provided a further boost to 0.56, while the more detailed Step-by-Step (SbS) CoT was significantly more effective, raising the score to 0.59. Our best fine-tuned system, which applied a self-refinement loop to the CoT-enhanced model's outputs, achieved a final Pass@1 of 0.62.

In parallel, we evaluated Gemini 2.5 Pro, a state-of-the-art proprietary model. A single-pass generation using a robust few-shot prompt achieved a score of 0.84. Applying our self-refinement loop to this model yielded our overall best result of **0.85**, which placed our team 4th on the official competition leaderboard.

6 Discussion

6.1 Error Analysis

A qualitative analysis of the failures reveals systematic challenges in both model reasoning and dataset construction. We categorize these errors into four primary themes, with detailed examples for each presented in Appendix C.

6.1.1 Ambiguous Instruction

A significant portion of errors originated from ambiguous or misleading problem statements. **Semantic ambiguity**, where the instruction was underspecified (Appendix C.1); **misleading instructions**, where the prompt suggested a simple algorithm but the test case required a more complex

System / Method	Training Data	Pass@1
Baseline (Qwen3)	trial (English)	0.50
Baseline (Qwen3)	trial (Bangla)	0.51
+ Data Augmentation	trial + verified_dev	0.54
+ CoT Fine-Tuning	trial + verified_dev + CoT (Hint)	0.56
+ CoT Fine-Tuning	trial + verified_dev + CoT (SbS)	0.59
+ CoT Fine-Tuning (Self-Refinement)	**trial + verified_dev + CoT (SbS)**	**0.62**
Gemini 2.5 Pro (Single-Pass)	N/A (Few-Shot)	0.84
Gemini 2.5 Pro (Self-Refinement)	**N/A (Few-Shot)**	**0.85**

Table 1: Pass@1 Performance of different systems on hidden **test set**.

one (Appendix C.2); and poor **translation quality**, which introduced ambiguity (Appendix C.3).

6.1.2 Failures of Constraint Adherence

Another major failure mode occurred when a model understood the general problem but failed to adhere to crucial, explicit constraints. This manifested as **test case ignorance**, where models, particularly smaller fine-tuned ones, prioritized a standard algorithm over the specific logic demanded by the test case (Appendix C.4). It also appeared as **memorization bias**, where models defaulted to common pre-trained patterns (e.g., a harmonic sum to n) instead of following a specific constraint (a sum to $n-1$) in the prompt (Appendix C.5).

6.1.3 Algorithmic Deficiencies

These errors represent Test case overfitting and reasoning failures. Models often produced hard-coded solutions that passed the visible test case but lacked generalizability (Appendix C.6); employed **sub-optimal or greedy algorithms** that failed in hidden test cases (Appendix C.7); and exhibited **semantic inconsistency**, resulting in runtime errors (Appendix C.8).

6.1.4 Inherent Dataset Challenges

Finally, a notable number of failures originated from structural flaws within the dataset itself: **Incorrect ground truth** in the test cases (Appendix C.9); **Incorrect test syntax** that would not be evaluated as intended by a standard Python interpreter (Appendix C.10); **Floating-Point Precision** issues (Appendix C.11); and **conflicting function signatures** between the instruction's example and the test case (Appendix C.12).

6.2 Findings

Our experiments yield several key insights. First, while the "silver-to-gold" augmentation consistently improved performance, translating the dataset back to English slightly degraded the results (Table 3), probably due to loss of semantic fidelity during the round-trip translation. This suggests that for our base model, which was already pre-trained on a large Bangla corpus, in-domain language consistency was more critical than alignment with its primary English pre-training. Second, our CoT experiments revealed a clear hierarchy of reasoning. The superior performance of Step-by-Step (SbS) plans over abstract hints indicates that models benefit more from explicit, structured problem decomposition. We conclude that SbS-style CoT provides a more effective learning signal, forcing the model to internalize a repeatable algorithmic workflow. Third, self-refinement enabled the model to reflect on execution errors and perform targeted repair, helping it resolve difficult edge cases that single-pass generation missed. Finally, as detailed in Section 6.1, many failures originated from the model's misinterpretation of ambiguous or misleading instructions, suggesting that clearer problem specifications could yield significant performance gains.

7 Limitations

Although TriGen achieves strong results, several limitations remain. The training data is relatively small, and part of it depends on silver-to-gold augmentation using a proprietary model, which may introduce bias. While the step-by-step reasoning and self-refinement loop improve performance, they rely heavily on the quality of the unit tests. When tests are incomplete, ambiguous, or contain errors, the refinement process can still converge on incorrect but test-passing solutions. Finally, the pipeline is optimized for single-function tasks, so its generalizability to more complex and diverse programming tasks has not yet been evaluated.

Conclusion

In this paper, we presented our TriGen system for Bangla prompt to Python code generation. Our experiments showed that combining data augmentation, chain-of-thought reasoning, and self-refinement led to our best fine-tuned system, while Gemini with few-shot prompting and self-refinement achieved the strongest overall result. Our analysis also reveals that limited training data and instruction ambiguity still constrain system reliability. In addition, our CoT supervision was generated automatically with Gemini and not manually validated. Even partial human inspection could provide higher-quality signals. In future work, we aim to expand high-quality Bangla code datasets, explore richer test case contexts during training and refinement to improve generalization, and extend TriGen to other low-resource languages.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tai, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *J. Mach. Learn. Res.*, 25(1).

Zhaopeng Feng, Yan Zhang, Hao Li, Bei Wu, Jiayu Liao, Wenqiang Liu, Jun Lang, Yang Feng, Jian Wu, and Zuozhu Liu. 2024. Tear: Improving llm-based machine translation with systematic self-refinement. *arXiv preprint arXiv:2402.16379*.

Andres Gonzalez, Md Zobaer Hossain, and Jahedul Alam Junaed. 2024. Numdecoders at semeval-2024 task 7: Flant5 and gpt enhanced with cot for numerical reasoning. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1260–1268.

Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices*, 46(1):317–330.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, and 1 others. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Sathvik Joel, Jie JW Wu, and Fatemeh H Fard. 2024. A survey on llm-based code generation for low-resource and domain-specific programming languages. *arXiv preprint arXiv:2410.03981*.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. 2024. Lisa: Reasoning segmentation via large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9579–9589.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Omar Faruqe Riyad, Trina Chakraborty, and Abhishek Dey. 2023. Team_Syrax at BLP-2023 task 1: Data augmentation and ensemble based approach for violence inciting text detection in Bangla. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 247–254, Singapore. Association for Computational Linguistics.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Peng Wang, Songshuo Lu, Yaohua Tang, Sijie Yan, Wei Xia, and Yuanjun Xiong. 2024. A full-duplex speech dialogue scheme based on large language model. *Advances in Neural Information Processing Systems*, 37:13372–13403.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2023. Multilingual machine translation with large language models: Empirical results and analysis. *arXiv preprint arXiv:2304.04675*.

A Performance on Dev-set and Language Impact

Model	Pass@1
TigerLLM-1B-it	0.15
Llama-3.2-3B-Instruct	0.25
Qwen2.5-Coder-3B-Instruct	0.51
Qwen3-4B-Instruct	0.58

Table 2: Pass@1 performance of different instruction-tuned models on the **dev-set**.

Model	Dataset	English	Bangla
Llama3	Dev-set	0.18	0.25
Qwen3	Test-set	0.50	0.51
Gemini 2.5 Pro	Test-set	0.82	0.85

Table 3: Impact of language (English vs. Bangla) on Pass@1 performance across models trained on trial set.

Table 2 shows Pass@1 performance of different models on the dev-set. Table 3 shows impact of language (English vs. Bangla) on Pass@1 performance.

B Dataset

The dataset was provided in JSON format, where each object contained a unique identifier, a Bangla instruction, an optional reference solution, and a list of test cases. Table 4 presents a sample entry, while Table 5 summarizes the number of instances in each set.

C Error Analysis - Examples

This appendix provides detailed examples for each category of error discussed in Section 6.1.

C.1 Semantic Ambiguity

Example 1

- **ID:** 15 (dev-set)
- **Instruction:** "একটি প্রদত্ত টুপল টুপল এর সংখ্যার গড় মান খুঁজে পেতে একটি ফাংশন লিখুন।"

Field	Example
id	231
instruction	প্রদত্ত অ্যারে থেকে সমান উপাদান জোড়া গণনা করার জন্য একটি পাইথন ফাংশন লিখুন।

response	`def count_Pairs(arr,n):`
	`  cnt = 0;`
	`  for i in range(n):`
	`    for j in range(i + 1,n):`
	`      if (arr[i] == arr[j]):`
	`        cnt += 1;`
	`  return cnt;`

test_list	['assert count_Pairs([1,1,1,1],4) == 6', 'assert count_Pairs([1,5,1],3) == 1', 'assert count_Pairs([3,2,1,7,8,9],6) == 0']

Table 4: An example entry from the Bangla code generation dataset.

Dataset	Number of Data
trial-set	74
dev-set	400
verified_dev	392
trial + verified_dev	466
test-set	500

Table 5: Distribution of datasets used in our experiments.

- **Analysis:** The instruction is ambiguous. It could refer to an overall average, row-wise averages, or column-wise averages. Only the test case clarified that column-wise averages were required.

Example 2

- **ID:** 66 (test-set)
- **Instruction:** "একটি আয়তক্ষেত্রে বর্গক্ষেত্রের সংখ্যা গণনা করার জন্য একটি পাইথন ফাংশন লিখুন।"
- **Model Response:**

~~~~
```python
def count_Squares(m,n):
    return (m*n)
```
~~~~

- **Analysis:** The problem is naturally interpreted as "how many 1×1 squares fit". That leads to m * n. But the test case clarified that it's about counting all possible squares (of all sizes).

C.2 Misleading Instruction

- **ID:** 5 (test-set)
- **Instruction:** "একটি স্ট্রিংকে ছোট অক্ষরে বিভক্ত করার জন্য একটি ফাংশন লিখুন।"
- **Analysis:** The instruction suggests we should just separate lowercase letters. However, the test case `split_lowerstring("AbCd") == ['bC', 'd']` reveals that a more complex logic is needed, where each new substring **starts** with a lowercase letter.

C.3 Translation Quality

Example 1

- **ID:** 15 (test-set)
- **Instruction:** "একটি প্রদত্ত অ্যারেতে পুনরাবৃত্তি না হওয়া উপাদানগুলির পণ্যটি খুঁজে পেতে একটি পাইথন ফাংশন লিখুন।"
- **Analysis:** The data point contains incorrect translation, where "product" was translated to 'পণ্য'. It has no contextual meaning.

Example 2

- **ID:** 67 (test-set)
- **Instruction:** "একটি পাইথন ফাংশন লিখুন যাতে সম এবং অদ্ভুত অঙ্কগুলির যোগফলের মধ্যে পার্থক্য খুঁজে পাওয়া যায়।"
- **Analysis:** The literal translation is incorrect in the mathematical context, where 'জোড়' and 'বেজোড়' should have been used instead of 'সম' and 'অদ্ভুত'.

C.4 Test Case Ignorance

- **ID:** 91 (test-set)
- **Instruction:** "প্রদত্ত অ্যারেতে k-তম উপাদান খুঁজে পেতে একটি ফাংশন লিখুন।"
- **Test Case:**

```
assert kth_element
([12,3,5,7,19], 5, 2) == 3
```

- **Analysis:** Models often defaulted to the standard "k-th smallest" algorithm (which would yield 5). They failed to adhere to the test case, which specified a positional lookup (the element at the second position of the unsorted array is 3).

C.5 Memorization Bias

- **ID:** 238 (test-set)

- **Instruction:** “n-1 এর হারমোনিক সমষ্টি গণনা করার জন্য একটি ফাংশন লিখুন।”

- **Model Response:** The model generated code to calculate the harmonic sum of ‘n’.

- **Analysis:** The model defaulted to the most common version of the harmonic sum problem, ignoring the specific ‘n-1’ constraint in the prompt. It defaulted to the patterns seen during pretraining.

C.6 Overfit Logic

- **ID:** 338 (test-set)

- **Instruction:** “একটি ফাংশন লিখুন যা প্রদত্ত দৈর্ঘ্যের ক্রমগুলি গণনা করে যার অ-নেতিবাচক উপসর্গ যোগফল রয়েছে যা প্রদত্ত মান দ্বারা উত্পন্ন হতে পারে।”

- **Test Case:** ‘assert bin_coff(4) == 2’

- **Model Response (Qwen3):**

~~~
```python
def bin_coff(n):
    if n <= 0: return 0
    if n == 1: return 1
    return 2
```
~~~

- **Analysis:** The smaller fine-tuned model generated a hardcoded solution that passes the single public test case but contains no generalizable algorithm.

C.7 Sub-optimal Algorithm Choice (Greedy Approach)

- **ID:** 29 (test-set)

- **Instruction:** “একটি প্রদত্ত স্ট্রিং এর অক্ষরগুলোকে পুনরায় সাজানো যায় কিনা তা পরীক্ষা করার জন্য একটি ফাংশন লিখুন যাতে একে অপরের সাথে সংলগ্ন দুটি অক্ষর ভিন্ন হয়।”

- **Analysis:** This problem requires careful handling of character frequencies to avoid getting ‘stuck.’ The optimal solution often involves a max-heap to prioritize placing the most frequent characters first. Our fine-tuned models often defaulted to a simpler but incorrect greedy approach that failed on more complex hidden test cases (e.g., “aaabc”).

C.8 Example: Semantic Error

- **ID:** 116 (test-set)

- **Instruction:** “দুটি প্রদত্ত সংখ্যার সাধারণ বিভাজকগুলির যোগফল খুঁজে বের করার জন্য একটি পাইথন ফাংশন লিখুন।”

```
def sum(a, b):
    # your code
    return a
```

- **Analysis:** The model generated a function named ‘sum’, which shadows the built-in Python ‘sum()’ function. The subsequent call to ‘sum()’ inside the function now refers to the function itself, not the Python built-in, causing an infinite recursion and a ‘RecursionError’ during execution. This represents a failure to consider the broader context of the programming language’s standard library.

C.9 Incorrect Ground Truth

- **ID:** 451 (test-set)

- **Test Case:**

```
assert upper_ctr('PYthon')
    == 1
```

- **Analysis:** The expected count of uppercase letters is incorrect (should be 2).

C.10 Incorrect Test Syntax

- **IDs:** 303, 426 (test-set)

- **Test Cases:**

```
assert pos_nos([-1,-2,1,2])
    == 1,2

assert neg_nos([-1,4,5,-6])
    == -1,-6
```

- **Analysis:** The right-hand side of these assertions is not a valid tuple. Python’s ‘assert’ syntax is ‘assert expression, [message]’. Consequently, the test ‘assert pos_nos(...) == 1,2’ is interpreted as ‘assert (pos_nos(...) == 1), 2’, where ‘2’ is the optional error message. This means the test would incorrectly pass if the function returned ‘1’. The correct syntax should have enclosed the expected output in parentheses, e.g., ‘== (1, 2)’.

C.11 Floating-Point Precision Issues

- **ID:** 129 (test-set)
- **Test Case:**

```
    assert circle_circumference(10)
        == 62.830000000000005
```

- **Analysis:** The test case expects a very specific floating-point number. A solution using Python's more accurate ‘math.pi‘ would fail this test. This forces the model to reverse engineer a less accurate hardcoded value for π (e.g., 3.1415) to pass the test, penalizing standard and correct programming practices in favor of a specific floating-point representation.

C.12 Conflicting Function Signatures

- **ID:** 338 (test-set)
- **Instruction's Example Signature:**

```
def bin_coff(n, r):
    # your code
    return n
```

- **Test Case:**

```
    ['assert bin_coff(4) == 2']
```

- **Analysis:** The instruction provides an example function signature with two parameters (‘n’, ‘r’), while the test case invokes the function with only a single argument (‘n=4’). This creates a direct conflict that the model must resolve. A model that incorrectly prioritizes the instruction's example would generate a two-parameter function, leading to an immediate ‘TypeError’ at runtime when the single-argument test case is executed.

D Data Pre-processing and Augmentation Details

This section describes the data pre-processing, augmentation, and formatting pipelines used in our experiments.

D.1 Initial Data Cleaning and Normalization

Before any training, we applied several cleaning steps to the raw ‘trial’, ‘dev’, and ‘test’ datasets.

- **Newline Normalization:** We observed that some entries contained Windows-style newline characters (\r\n). All newlines were standardized to the Unix-style (\n) to prevent the model from learning and reproducing inconsistent line breaks.
- **Code Fence Enforcement:** To ensure the model learned the precise output format, we programmatically verified that every response in our training data was correctly enclosed in a ```python ... ``` code fence. Any responses missing these fences were automatically wrapped.

D.2 “Silver-to-Gold” Data Augmentation

To augment our training data, we generated a high-quality, verified version of the ‘dev’ set.

1. **Silver Data Generation:** We used the Gemini 2.5 Pro API to generate solutions for all problems in the ‘dev’ set, creating our initial “silver” dataset.
2. **Execution-Based Verification:** We then executed each generated solution against its corresponding unit tests.
3. **Gold Data Filtering:** Only the solutions that passed the test case were retained, resulting in a high-fidelity ‘gold’ or ‘verified_dev’ set. This set was then merged with the original ‘trial’ set to form our final augmented training corpus.

D.3 Translation

To investigate the impact of language, we translated all Bangla instructions into English. For translations, we compared the output of the Googletrans library with the Gemini 2.5 Pro API. Upon manual inspection, we found the Gemini-generated translations to have higher semantic fidelity and contextual accuracy. All final translation experiments were therefore conducted using the Gemini-translated datasets.

D.4 Dynamic Prompt Construction for Fine-Tuning

we constructed the prompts during the data loading phase to serve as a form of data augmentation and to provide richer context to the model. Our prompt construction function performed the following steps for each training example:

1. **Function Signature Extraction:** The public 'test_list' was parsed to programmatically extract the correct function name and signature. This information was explicitly added to the prompt to mitigate errors from conflicting or ambiguous signatures in the original instruction.

2. **Final Formatting:** The dynamically constructed user prompt was then formatted using the specific chat template of the model being trained (e.g., Qwen3 or Llama 3) to ensure proper tokenization and handling of special roles like 'system', 'user', and 'assistant'.

Finally, during the training process itself, we employed a loss mask to ensure the model only learned from the assistant's response tokens, ignoring the prompt tokens. This focuses the model's learning entirely on the target output.

E Experimental Setup

This section provides a detailed overview of the models, hyperparameters, and infrastructure used for our fine-tuning and inference experiments.

E.1 Fine-Tuning Infrastructure and Model Configuration

All fine-tuning experiments were conducted on a single T4 GPU with 16GB of VRAM, provided via Google Colab. To facilitate training on this hardware, we leveraged the `unsloth` library for memory-efficient model loading and optimization.

The base model for our fine-tuning experiments was `Qwen/Qwen3-4B-Instruct`. It was loaded in 8-bit precision with a maximum sequence length of 1024 tokens. We then applied Parameter-Efficient Fine-Tuning (PEFT) using the LoRA (Low-Rank Adaptation) methodology with the following configuration, as shown in Table 6.

LoRA Parameter	Value
Rank (`r`)	16
Alpha (`lora_alpha`)	32
Dropout (`lora_dropout`)	0.0
Bias	`none`

Table 6: LoRA configuration used for fine-tuning experiments.

E.2 Training Hyperparameters

We used the `SFTTrainer` from the TRL library for supervised fine-tuning. To prevent overfitting and select the best model checkpoint, we split our training data into a 90% training set and a 10% validation set. We enabled early stopping with a patience of 3 epochs, monitoring the `eval_loss` on the validation set. All models were trained for a maximum of 10 epochs. The key training hyperparameters are detailed in Table 7.

Hyperparameter	Value
Learning Rate	5×10^{-5}
Batch Size (per device)	4
Effective Batch Size	8
Optimizer	AdamW (8-bit)
LR Scheduler	Cosine
Warmup Ratio	0.1
Weight Decay	0.01
Precision	FP16

Table 7: Key hyperparameters used for the SFTTrainer.

E.3 Inference Strategy

E.3.1 Fine-Tuned Model Inference

For generating solutions from our fine-tuned models, we employed a batched generation strategy to maximize throughput. The tokenizer's padding side was set to 'left' to ensure correct output in a batch context. We used deterministic decoding by setting 'do_sample=False'. A batch size of 8 was used, with a maximum generation length of 1024 tokens.

E.3.2 API-Based Model Inference

For our experiments with the Gemini 2.5 Pro API, we developed a separate inference script. This script processed prompts sequentially but included a retry mechanism with exponential backoff to handle API rate limits and errors. To ensure persistence and prevent data loss, results were saved to a JSONL file after each successful API call, making the process fully resumable. The generation was performed with a low temperature of 0.1 to favor deterministic and correct code.

E.4 Evaluation

The official evaluation metric for this shared task is the **pass rate**, also referred to as **Pass@1**. This metric is defined as the percentage of problems for which a system's generated code passes all hidden unit tests when executed in a sandboxed environment. A higher pass rate corresponds to a higher rank on the competition leaderboard. All scores reported in this paper uses **Pass@1**.

F Prompts Used for Generation and Refinement

This section details the system and user prompts employed for our experiments.

F.1 System Prompt for Initial Code Generation

Figure 2 shows the system prompt that was used for the "Think" (initial generation) stage. It is designed to enforce strict output formatting and guide the model's logic.

F.2 User Prompt Template for Initial Generation

The user prompt was dynamically constructed for each problem, providing the instruction and the public test case as structured input (Figure 3).

F.3 System Prompt for Self-Refinement

For the "Refine" (self-correction) stage, a different system prompt was used to frame the task as a debugging and code review exercise (Figure 4).

F.4 User Prompt Template for Self-Refinement

The user prompt for the refinement loop was dynamically constructed to include the execution feedback (Figure 5).

~~~
You are an expert, competition-focused Python programmer.
Your task is to generate lean, correct, and executable Python code to solve the given problem.

**Core Requirements:**
- Your entire response MUST be a single, fenced code block starting with ```python and ending with
    ```.
- Only generate the function or class requested. Do not include example usage or print statements.
- The code must be self-contained and runnable.
- The function must use the exact name and signature from the provided instruction or test cases.
- If helper classes are required by the test cases, you must define them.

**Code Style Requirements:**
- **No Explanations:** Do not add comments that explain your reasoning, translate the instruction, or
     describe basic Python functionality.
- **Minimal Docstrings:** If you include a docstring, it MUST be a single line explaining the
    function's high-level purpose. Do not use multi-line docstrings or describe arguments (Args/
    Returns).
- **Example of desired style:**
  ```python
  def add_numbers(a, b):
      \"\"\"Returns the sum of two numbers.\"\"\"
      return a + b
  ```

**Logic Requirements:**
- If the instruction is ambiguous, the provided test cases are the source of truth.
- Prioritize correctness and efficiency. Do not generate pseudo-code.
~~~

Figure 2: System Prompt

````
{instruction}

You must implement the solution strictly following the function signature in the instruction. If any
    helper classes or methods are needed to run the following test cases, you must define them as
    well.

Test cases:
{test_list}
````

Figure 3: User prompt

~~~
You are an expert Python code reviewer and debugger.
Your task is to fix a flawed Python function that failed to solve a programming problem.

You will be presented with:
1.  The original problem description.
2.  Your previous, incorrect code submission.
3.  The specific error message or failed test case that caused the failure.

**--- Your Goal: Analyze and Correct ---**
- **Analyze the Error:** Carefully examine the failed test case or error message. This is the most
    important clue to understanding the mistake.
- **Identify the Flaw:** Compare the failed test case with your incorrect code to pinpoint the
    logical flaw.
- **Implement the Fix:** Write a new, corrected version of the Python code that directly addresses
    the identified flaw and will pass the test case.
- **Generalize:** Ensure the corrected solution is robust and handles other potential edge cases, not
     just the single failed test.

**--- Output Requirements ---**
- Your entire response MUST be a single fenced code block beginning with ```python and ending with
    ```.
- Do not include any text, explanations, or apologies before or after the code block.
- Provide only the corrected, complete, and self-contained Python code.

**Example of the ONLY acceptable output format:**
  ```python
  def add_numbers(a, b):
      \"\"\"Returns the sum of two numbers.\"\"\"
      return a + b
  ```
~~~

Figure 4: Self-refinement System prompt

````
**Problem:**
{instruction}

**Your Incorrect Code:**
{failed_code}

**Reason for Failure:**
{error_info}

Based on the error, provide a corrected and complete Python code
that solves the problem and passes the failed test case.
````

Figure 5: Self-refinement user prompt

TeamB2B at BLP-2025 Task 2: BanglaForge: LLM Collaboration with Self-Refinement for Bangla Code Generation

Mahir Labib Dihan, Sadif Ahmed, Md Nafiu Rahman
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka, Bangladesh
`{mahirlabibdihan, ahmedsadif67, nafiu.rahman}@gmail.com`

Abstract

Bangla is a low-resource language for code generation, lacking large-scale annotated datasets and tools to transform natural language specifications into executable programs. This makes Bangla-to-code generation a challenging task requiring innovative solutions. To address this, we introduce **BanglaForge**, a novel framework for generating code from Bangla function descriptions. BanglaForge leverages a retrieval-augmented dual-model collaboration paradigm with self-refinement, combining in-context learning, llm-based translation, systematic prompt engineering, and iterative self-refinement based on execution feedback, where a coder generates initial solutions and a reviewer enhances them for robustness. On the **BLP-2025 Bangla Code Generation** benchmark, BanglaForge achieves a competitive Pass@1 accuracy of **84.00%**, demonstrating the effectiveness of retrieval, model collaboration, and self-refinement for low-resource Bangla code generation.

1 Introduction

Large language models (LLMs) have shown strong capabilities in code generation, where natural language descriptions are automatically transformed into executable programs. Models such as Codex, CodeT5, and StarCoder, trained on large-scale codetext corpora, can produce syntactically valid and semantically correct solutions, performing well on benchmarks like HumanEval (Chen et al., 2021). These advances reduce the gap between human intent and code, making programming more accessible. However, most existing systems are designed for English inputs, leaving low-resource languages underserved. Models often struggle with informal structures, domain-specific terms, and semantic nuances, resulting in incorrect or brittle outputs.

We introduce **BanglaForge**, a framework for generating executable code from Bangla task descriptions. Each input is represented as a triple: the Bangla description, its English translation, and unit test assertions. This structure leverages the models stronger English understanding while retaining Bangla context. BanglaForge combines retrieval-augmented prompting, iterative self-refinement with execution feedback, and a dual-model coderreviewer pipeline. Our system achieves a **Pass@1 accuracy of 84%** on **BLP-2025 Bangla Code Generation Benchmark** (Raihan et al., 2025c), demonstrating the potential of practical low-resource code generation.

Our contributions can be summarized as follows:

- A retrieval-augmented few-shot prompting approach using TF-IDF to select relevant Bangla–Python pairs, improving in-context learning despite limited labeled data.
- A LLM-based translation component that translates Bangla instructions into English with the help of a glossary to enable accurate cross-lingual code generation.
- An iterative self-refinement protocol that leverages execution feedback to detect and correct errors across refinement cycles.
- A dual-model architecture where a generator model focuses on functional correctness and a reviewer model enhances robustness, style, and coverage of edge cases.

We release our implementation of BanglaForge at `https://github.com/mahirlabibdihan/BanglaForge` to facilitate reproducibility and further research.

2 Related Works

Research in Bangla NLP has evolved from early word embeddings to specialized LLMs. Initial efforts such as BnVec introduced embeddings

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 642–655
December 23, 2025 ©2025 Association for Computational Linguistics

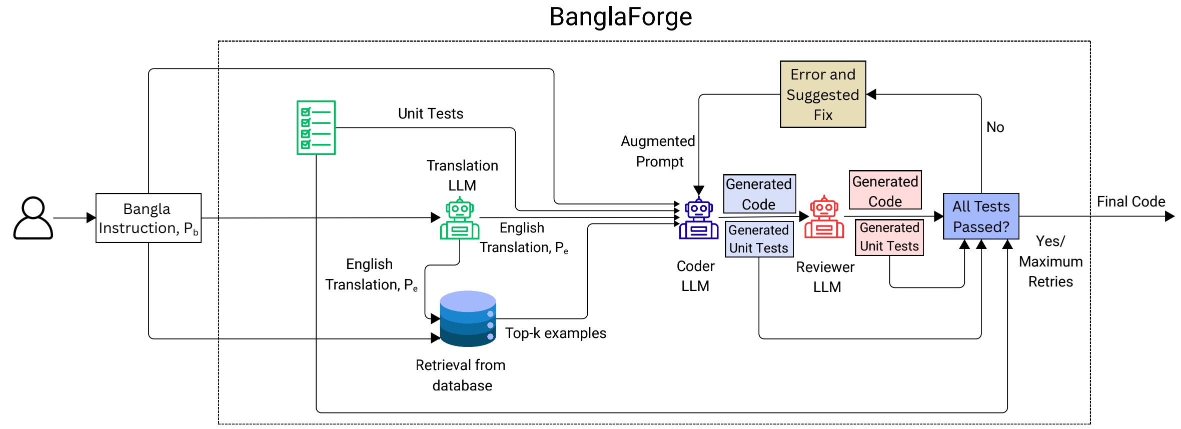


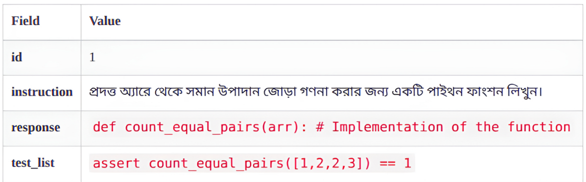
Figure 1: Workflow of the proposed **BanglaForge** framework. A Bangla instruction (P_b) is translated into English (P_e) and, together with unit tests, used to retrieve top-k bilingual examples. The **Coder LLM** then generates Python code and additional test cases. The **Reviewer LLM** validates, refines, and re-prompts upon errors until all tests (original and generated) are passed, yielding the final code.

like fastText, Word2Vec, and GloVe trained on diverse corpora, with customized fastText outperforming multilingual baselines in classification tasks (Kowsher et al., 2021, 2022; Mojumder et al., 2020). Recent advances include Bangla LLMs and benchmarks such as TigerCoder (Raihan et al., 2025b) and BanglaByT5 (Bhattacharyya and Bhattacharya, 2025), which advanced code generation and tokenization strategies. However, existing work largely focuses on pretraining and benchmarking without complete generation pipelines. Our work addresses these gaps by introducing retrieval-augmented prompting, iterative self-refinement, and a dual generator-reviewer design. A detailed discussion is provided in Appendix A.

3 Dataset

We build on the resources introduced for Bangla code generation across recent shared tasks and benchmarks. Our dataset comes from the Bangla Code Generation shared task (Task 2) at BLP-2025 (Raihan et al., 2025c), where the objective is to translate Bangla natural language programming prompts into Python functions that satisfy hidden unit tests. The dataset is distributed through an official starter kit[1], which also provides baseline code and evaluation scripts.

Each entry is a JSON object containing four fields: an `id`, a Bangla instruction describing the task, a `response` field with the reference Python implementation (training only), and a `test_list` field of assert-based unit tests.



Field	Value
id	1
instruction	প্রদত্ত অ্যারে থেকে সমান উপাদান জোড়া গণনা করার জন্য একটি পাইথন ফাংশন লিখুন।
response	`def count_equal_pairs(arr): # Implementation of the function`
test_list	`assert count_equal_pairs([1,2,2,3]) == 1`

Figure 2: Example data point

Split	Purpose	Size
Trial	Initial experiments	74
Development	Validation	400
Test	Final evaluation	500

Table 1: Dataset Split Statistics for Bangla Code Generation

For development and testing, we adopt two external Bangla code generation benchmarks. The **mHumanEval-Bangla** dataset (Raihan et al., 2025a), a Bangla extension of HumanEval, is used during the development phase, enabling programmatically testable evaluation on held-out prompts. The **MBPP-Bangla** dataset (Raihan et al., 2025b), adapted from MBPP as part of the TigerCoder framework, is used during both development and test phases, providing diverse programming problems in Bangla with associated unit tests.

4 Methodology

We propose **BanglaForge**, a retrieval-augmented dual-LLM framework for generating Python code from Bangla natural language specifications.

[1] https://noshinulfat.github.io/blp25_code_generation_task/#/get-started

The system tackles low-resource code generation through structured prompt design, bilingual translation, example retrieval, and a two-stage generation-review process involving a *Coder LLM* and a *Reviewer LLM*. Together, these components ensure both functional correctness and stylistic reliability, even in underrepresented languages like Bangla. An overview of the complete workflow is shown in Figure 1 and in Algorithm 1 (Appendix). Each stage is described in detail below.

4.1 Problem Formulation and Input Representation

Each task in the dataset consists of a Bangla instruction P_b and its corresponding public unit tests $T = \{t_1, \ldots, t_n\}$. To enable code synthesis, the instruction is translated into English using a translation model equipped with a controlled glossary for mathematical and algorithmic terms (e.g., GCD, LCM, sum). The glossary is curated by the authors which was motivated from the provided dataset and commonly seen technical terms in code related works. The translated instruction P_e retains the semantic fidelity of the Bangla instruction while ensuring syntactic clarity for code generation. The systems objective is to synthesize a Python function f such that all $t_i \in T$ are satisfied given the constraints in P_b. Function prototypes are normalized to valid Python syntax, aligning argument and return types with unit test definitions.

4.2 Retrieval-Augmented Example Selection.

To enhance contextual understanding, both Bangla and English task descriptions are used to retrieve semantically similar solved examples from a bilingual database $\mathcal{D} = \{(p_i^b, p_i^e, c_i, T_i)\}_{i=1}^N$. Each entry contains the Bangla and English prompts, the reference code (c_i), and associated test cases (T_i). Both P_b and P_e are embedded using TF-IDF unigram bigram representations. We chose TF-IDF due to its high computational efficiency and strong performance on smaller datasets, as dense retrievers typically require a large training corpus to be effective (Arabzadeh et al., 2021). For our task, TF-IDF's strength in matching exact, high-signal technical keywords (e.g., GCD," factorial") is paramount. This lexical precision provides a fast and more reliable baseline for retrieving analogous code problems than a dense model's generalized semantic understanding (Karpukhin et al., 2020). The top-k examples (typically $k = 5$) are selected and inserted into the prompt as few-shot exemplars. For experiments on the Development set, the database $\mathcal{D}$ consists of the Trial set. For experiments on the Test set, we use the combined Trial+Development sets as the database. This bilingual, retrieval-augmented setup enables contextual grounding and helps the model capture problem-solving patterns from similar tasks. The retrieved example format is provided in Appendix C.

4.3 Stage 1: Code Generation by Coder LLM.

The *Coder* LLM receives a composite input consisting of the Bangla instruction P_b, English translation P_e, the retrieved top-k example pairs (p_i^b, p_i^e, c_i, T_i), and the provided unit tests T. Based on this augmented prompt, the Coder LLM generates a Python code candidate c_0 intended to satisfy T, and additional synthetic test cases T_c designed to cover potential edge or missing cases. This stage focuses on functional code generation guided by contextual analogies from retrieved examples. The output (c_0, T_c) is then passed to the Reviewer LLM for refinement. The detailed prompt for Coder LLM is provided in Appendix C.

4.4 Stage 2: Code Review and Refinement by Reviewer LLM.

The *Reviewer* LLM acts as a validator and refiner. It takes as input the code and test cases generated by the Coder LLM along with the original task description and unit tests. Its responsibilities include running static and logical checks on c_0, correcting syntax or runtime issues, improving variable naming, structure, and input validation, generating an additional set of refined unit tests T_r to ensure covering edge cases. If any error or inconsistency is detected, the Reviewer LLM suggests an explicit fix and the process is repeated up to a maximum of M iterations ($M = 5$). The detailed prompt for Reviwer LLM is provided in Appendix C.

4.5 Iterative Self-Refinement Protocol.

The refinement loop is formally defined as: $c_{i+1} = \mathcal{R}(c_i, e_i, \mathcal{P})$, where $\mathcal{R}$ denotes the Reviewer LLM, e_i is the detected error, and $\mathcal{P}$ represents the augmented prompt containing feedback. The cycle continues until all test casesoriginal (T), coder-generated (T_c), and reviewer-generated (T_r)are successfully passed, or until the retry limit M is

Model	Few Shot	# Examples	Translation	# Unit Tests	Pass@1
Dev Set					
Gemma-1B	N/A	0	No	0	27.25%
GPT-OSS-20B	Manual	3	No	0	60.25%
GPT-OSS-20B	Manual	5	No	0	61.25%
DeepSeek-R1-Llama-70B	Manual	5	Yes	0	57.75%
Gemini-2.0-Flash	Manual	3	Yes	0	60.00%
Gemini-2.0-Flash	Manual	5	Yes	0	62.50%
Lg Exaone Deep 32B	Manual	5	Yes	1	85.25%
Lg Exaone Deep 32B	Manual	5	Yes	3	94.25%
Lg Exaone Deep 32B	RAG (Trial)	5	Yes	3	95.50%
Test Set					
Lg Exaone Deep 32B	RAG (Trial+Dev)	5	Yes	1	80.60%
Gemini-2.5-Pro	**RAG (Trial+Dev)**	**5**	**Yes**	**1**	**84.00%**

Table 2: Pass@1 accuracy of models on the BLP-2025 Development and Test sets.

reached. This multi-level testing ensures that the final solution generalizes beyond the given test cases. The errors and suggested fixes are provided in Appendix C.

5 Experiment

5.1 Evaluation Metrics

We evaluate performance using the Pass@1 accuracy metric, which measures the proportion of problems solved correctly in the first iteration. This metric provides a clear and direct assessment of the systems accuracy in solving problems without requiring further refinements.

5.2 Models

We evaluate several large language models (LLMs) for Bangla code generation. The models tested on the Development set include **Gemma-1B** (Gemma, 2024), **GPT-OSS-20B** (Initiative, 2024), **DeepSeek-R1-Llama-70B** (AI, 2025), **Gemini-2.0-Flash** (DeepMind, 2024), and **Lg Exaone Deep 32B** (Research, 2024), with different prompting strategies and unit-test settings. For the final evaluation on the Test set, we select **Lg Exaone Deep 32B** (Research, 2024) and **Gemini-2.5-Pro** (DeepMind, 2025) under their best-performing configurations within a retrieval-augmented dual-stage pipeline.

5.3 Results

We evaluate our system on the BLP-2025 Bangla code generation benchmark. Our experiments are conducted in two stages: first on the Development set to explore different models and prompting strategies, and then on the Test set to report final results. Table 2 presents the Pass@1 accuracy for various models and configurations across both sets.

The development set results reveals that small-scale models such as **Gemma-1B** achieve only 27.25% Pass@1, underscoring the challenge of Bangla-to-code translation without contextual guidance. Larger open-source models like **GPT-OSS-20B** shows improvements (60.25-61.25%) under few-shot prompting, though performance gains taper off with additional in-context examples. Introducing translation-based prompting further improves comprehension of Bangla instructions, as seen with **DeepSeek-R1-Llama-70B** (57.75%) and **Gemini-2.0-Flash** (60-62.5%).

A major performance leap is observed with the **Lg Exaone Deep 32B** model, which combines translation and lightweight unit-test feedback. Accuracy rises from 85.25% with one visible test to 94.25% with three tests, highlighting the benefit of guided reasoning through intermediate validation. When enhanced with our RAG pipeline on the trial set, the model achieves 95.5% Pass@1 on the development benchmarkdemonstrating consistent improvements through contextual retrieval and refinement.

On the held-out test set, the RAG-augmented **Lg Exaone Deep 32B** achieves 80.6% Pass@1, while the more recent **Gemini-2.5-Pro** model further pushes performance to **84.0%**. These results confirm that retrieval augmentation combined with multilingual comprehension yields robust generalization across unseen Bangla programming tasks.

6 Conclusion

In this paper, we presented a retrieval-augmented dual-model framework for generating Python code from Bangla instructions. Combining structured prompting, iterative self-refinement, and a generator-reviewer design, our system achieved Pass@1 accuracy of 84% on the BLP-2025 benchmark. The approach consistently outperforms baselines, showing the effectiveness of retrieval augmentation and feedback-driven refinement for low-resource code generation. Future work will expand the framework to other languages and incorporate reinforcement-based refinement. Additionally, improvements in RAG corpus and Bangla-to-English translation quality are expected to further enhance the overall performance of the pipeline.

7 Limitations

While BanglaForge demonstrates strong performance on the BLP-2025 Bangla code generation benchmark, several limitations remain. First, the system relies heavily on high-quality bilingual translation; inaccuracies in Bangla-to-English mapping or glossary coverage can propagate errors to the generation stage. Second, the retrieval component depends on TF-IDF, which captures lexical overlap but may miss deeper semantic similarities, especially in complex algorithmic prompts. Third, the framework assumes well-structured Bangla input; informal phrasing or dialectal variations could reduce translation fidelity and retrieval relevance. Additionally, self-refinement cycles are limited to a fixed number of iterations and do not incorporate adaptive stopping or learning from prior refinements. Finally, since the dataset itself originates from machine-translated English sources, true Bangla-native problem framing and linguistic diversity remain under-represented. Future work should explore human-curated datasets, semantic retrieval models, and reinforcement-based refinement to address these limitations.

References

DeepSeek AI. 2025. Deepseek-r1: Reasoning models built on llama-70b. https://github.com/deepseek-ai/DeepSeek-R1. Accessed: 2025-10-05.

Negar Arabzadeh, Xinyi Yan, and Charles L. A. Clarke. 2021. Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection. *Preprint*, arXiv:2109.10739.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin 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. Association for Computational Linguistics.

Pramit Bhattacharyya and Arnab Bhattacharya. 2025. Banglabyt5: Byte-level modelling for bangla. *arXiv preprint arXiv:2505.17102*. Cs.CL.

Pramit Bhattacharyya, Joydeep Mondal, Subhadip Maji, and Arnab Bhattacharya. 2023. Vacaspati: A diverse corpus of bangla literature. *arXiv preprint arXiv:2307.05083*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Google DeepMind. 2024. Gemini 2.0: A family of multimodal language models. https://deepmind.google/technologies/gemini/. Accessed: 2025-10-05.

Google DeepMind. 2025. Gemini 2.5 pro: Next-generation multimodal reasoning model. https://deepmind.google/technologies/gemini-2-5/. Accessed: 2025-10-05.

Team Gemma. 2024. Gemma: Open models by google deepmind. https://deepmind.google/technologies/gemma/. Accessed: 2025-10-05.

Mohammad Mehadi Hasan, Fatema Binte Hassan, Md Al Jubair, Zobayer Ahmed, Sazzatul Yeakin, and Md Masum Billah. 2025. Bangla bert for hyperpartisan news detection: A semi-supervised and explainable ai approach. *arXiv preprint arXiv:2507.21242*.

Open Source Science Initiative. 2024. Gpt-oss: An open-source family of general-purpose large language models. https://huggingface.co/gpt-oss. Accessed: 2025-10-05.

Mohsinul Kabir, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Mir Tafseer Nayeem, M Saiful Bari, and Enamul Hoque. 2023. Benllmeval: A comprehensive evaluation into the potentials and pitfalls of large language models on bengali nlp. *arXiv preprint arXiv:2309.13173*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics.

Md. Kowsher, Md. Shohanur Islam Sobuj, Md. Fahim Shahriar, Nusrat Jahan Prottasha, and Mohammad Shamsul Arefin. 2022. An enhanced neural word embedding model for transfer learning. *Applied Sciences*, 12(6):2848.

Md. Kowsher, Md. Jashim Uddin, Anik Tahabilder, Nusrat Jahan Prottasha, Mahid Ahmed, K. M. Rashedul Alam, and Tamanna Sultana. 2021. Bnvec: Towards the development of word embedding for bangla language processing. *International Journal of Engineering and Technology*, 10(2):95–102.

Hemal Mahmud and Hasan Mahmud. 2024. Enhancing sentiment analysis in bengali texts: A hybrid approach using lexicon-based algorithm and pretrained language model bangla-bert. *arXiv preprint arXiv:2411.19584*.

Pritom Mojumder, Md. Faruque Hossain, Mahmudul Hasan, and K. M. Azharul Hasan. 2020. A study of fasttext word embedding effects in document classification in bangla language. In *ICONCS 2020*, pages 441–453. Springer, Cham.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Nishat Raihan and Marcos Zampieri. 2025. Tigerllm: A family of bangla large language models. *arXiv preprint arXiv:2503.10995*.

LG AI Research. 2024. Exaone deep 32b: Large-scale multilingual foundation model by lg ai research. `https://www.lgresearch.ai/model/exaone-deep-32b`. Accessed: 2025-10-05.

Sagor Sarker. 2020. Bangla-bert: A pretrained bert model for bengali. `https://github.com/sagorbrur/bangla-bert`.

Sheikh Shafayat, Quamran Hasan H. M. Minhajur Rahman Chowdhury Mahim, Rifki Afina Putri, James Thorne, and Alice Oh. 2024. Benqa: A question answering and reasoning benchmark for bengali and english. *arXiv preprint arXiv:2403.10900*.

A Related Works

The trajectory of research in Bangla NLP has shifted from foundational embeddings and lightweight classification models to full-fledged Bangla LLMs and, more recently, toward modular architectures that integrate retrieval and feedback. In the early days, emphasis was placed on crafting vector representations tailored to Banglas morphological richness and vocabulary distribution. The BnVec project, for instance, introduced Bangla-specific fastText, Word2Vec, and GloVe embeddings that placed importance on vocabulary coverage and representation quality (Kowsher et al., 2021). Later work showed that embeddings trained on Bangla corpora outperform multilingual embedding baselines in text classification and related tasks (Kowsher et al., 2022; Mojumder et al., 2020). Meanwhile, the Vacaspati corpus and derived models such as Vac-FT and Vac-BERT demonstrated that diversifying corpus domains and scaling data can boost embedding and language model utility beyond standard fastText baselines (Bhattacharyya et al., 2023).

As the field progressed, researchers began developing Bangla-centric pretrained language models for both understanding and generation. A notable early example is BanglaBERT, introduced by Bhattacharjee et al., which is a BERT (ELECTRA-discriminator)style model pretrained on a 27.5 GB Bangla corpus (Bangla2B+) and evaluated on a suite of Bangla NLU benchmarks that include classification, NLI, NER, and QA tasks under the BLUB benchmark (Bhattacharjee et al., 2022). BanglaBERT outperforms multilingual baselines on those tasks, showing that language-specific pretraining brings tangible gains in low-resource settings. Building on that, more recent works such as enhanced sentiment analysis pipelines fine-tune and hybridize BanglaBERT with lexicon/rule components (Mahmud and Mahmud, 2024), or apply it for domain tasks like hyperpartisan news detection with semi-supervised learning and explainability (Hasan et al., 2025). Alongside, general-purpose monolingual models for Bangla (e.g. Bangla-Bert-Base by Sagor Sarker et al.) have also been proposed and used across classification and NER tasks (Sarker, 2020).

Complementing these, newer model lines push toward generative and evaluation capacities in Bangla. TigerLLM, is a suite of Bangla LLMs trained on large Bangla corpora and shows gains over prior open and proprietary models across Bangla benchmarks (Raihan and Zampieri, 2025). In the programming domain, TigerCoder introduces dedicated Bangla code LLMs (1B and 9B) and the MBPP-Bangla benchmark, reporting 11 to 18 % Pass@1 improvement over multilingual baselines (?). In evaluation, BenLLMEval provides a wide evaluation of off-the-shelf LLMs (GPT-3.5, LLaMA-2, Claude, etc.) on Bangla tasks (summarization, QA, paraphrase, classification), revealing substantial performance gaps in zero-shot settings (Kabir et al., 2023). The BEnQA benchmark offers parallel BengaliEnglish QA and reasoning tasks derived from exam questions; it shows that chain-of-thought prompting helps reasoning tasks and that including English context can improve performance in Bengali (Shafayat et al., 2024).

Despite advances in modeling, most existing works treat the language model as a single-step generator without built-in mechanisms for grounding, correction, or iteration. In broader NLP and code domains, however, robust generation systems increasingly incorporate retrieval-augmented architectures (e.g. RAG), cross-lingual retrieval for low-resource grounding, retrievalaugmented data augmentation (RADA), multi-stage or hierarchical retrieval (e.g. for code), and iterative refinement via coderreviewer loops or test-driven feedback. These techniques have been shown to reduce hallucination, improve factual grounding, and correct logical or syntactic errors in generated outputs.

These retrieval, review, and iteration techniques remain underexplored in Bangla and especially in Banglacode generation. In this work, we explicitly address that gap by combining Bangla-focused models (e.g. TigerLLM, TigerCoder) with retrieval-based prompt augmentation, a separate reviewer module, and iterative self-refinement. This hybrid design aims to boost reliability and real-world usability in Bangla code generation systems.

B Experimental Setup

All models were configured with the following default generation parameters: `temperature` = 0.7, `top_p` = 0.9, and `max_new_tokens` = 1024. Each query generated `n` = 1 output sample per decoding pass.

C Model Prompts

This section details the prompts used in our **Bangla2Py** framework. The prompts are designed to guide the Large Language Models (LLMs) through the code generation, refinement, and review stages. Placeholders like `{instruction}` are dynamically populated by the pipeline.

C.1 Coder Model Prompts

The **Coder LLM** is the first stage of our system and is responsible for writing the initial Python solution. It receives both the Bangla task description and its English translation, along with a set of retrieved examples and the provided unit tests. The coder's system prompt clearly defines its role as a Python code generator and instructs it to produce only executable code no explanations or comments (Figure 3). The main task prompt includes several few-shot examples followed by the current problem. Each example shows the task instruction (in both languages), the correct solution, and unit tests (Figure 4). If the generated code fails any test, the coder receives a short feedback message describing the error type (e.g., syntax error, timeout, or assertion failure) along with a fix hint (Figure 5). It then regenerates an improved version in the next iteration. This feedback-guided prompting helps the coder LLM progressively refine its output and produce cleaner, test-ready code with a built-in `main()` function for validation.

C.2 Reviewer Model Prompts

The **Reviewer LLM** acts as the second stage and takes the code produced by the coder, along with the original BanglaEnglish instructions and all test cases (both given and generated). Its prompt defines the role of a code reviewer focusing on improving correctness, readability, and coverage of edge cases without changing the function signature. The reviewer checks for logical mistakes, inefficient loops, missing validations, or weak test coverage. It then returns a refined version of the code, adds extra corner-case tests, and ensures the final version passes both visible and hidden cases. If errors are still detected, the reviewer can repeat this process with updated feedback until all tests are passed or a retry limit is reached.

C.3 Few-Shot Example Template

To help both LLMs generalize better, we use retrieval-augmented few-shot examples in the

System Prompt for Coder Model

You are a Python programming assistant.
The user will provide a function stub where the original docstring is written in Bangla with a translated version and a unit test case.
Your task is to read the Bangla + English (Translated) docstring and the unit test case, understand the requirement, function parameters, return type, and complete the function implementation in Python.
Your response must be in English, not Bangla, and must only contain valid Python code.
Do not add explanations, comments, or extra text. Just return the code solution.
Your main task is to carefully read the Bangla + English (Translated) docstring and the unit test case and infer:
1. The expected number of parameter and their types
2. The expected return type
3. The correct implementation logic
Important guidelines:
1. The function signature is already provided in the instruction. Implement the function as specified.
2. Include a **main function** (using `def main():`) in your code that contains necessary unit tests or example calls to validate your function.
3. Do **not** call `main()` anywhere in your code. This will be executed externally.
4. Try to keep the code as simple as possible.
5. Your response should contain only one python block enclosed in a code block like:

```
“‘python
# your code here
“‘
```

Figure 3: System prompt for the coder model.

prompts. The system retrieves the top-k most similar problems from the bilingual database using both Bangla and English task texts. Each example includes:

- The Bangla and English instructions,
- The reference Python solution, and
- The corresponding unit tests.

These examples are formatted in a consistent template and placed before the current task in the prompt (see Figure 8). This structure lets the models recognize patterns in how Bangla instructions map to Python logic, guiding them to produce correct and well-structured code even for unseen problems.

D Error Refinement

The iterative feedback follows an augmentation protocol as outlined in Table 3.

E Ablation Study

To analyze the contribution of each component in **BanglaForge** , we perform ablation experiments using the **Lg Exaone Deep 32B** model on the BLP-2025 development set. The best full configuration achieves a Pass@1 accuracy of **95.5%**, and all reported variations are measured relative to this setting. Each ablation disables or modifies a single module while keeping the rest of the pipeline fixed.

Error Type	Feedback Hint / Guidance
Syntax Error	Check indentation, missing colons, or parentheses; ensure valid Python syntax.
Runtime Error	Ensure variables are initialized and referenced correctly; verify data types and control flow.
Assertion Failure	Compare expected vs. actual outputs; review logical steps and boundary conditions.
Timeout Error	Optimize loops or recursion; include clear termination conditions.
System Exit	Avoid abrupt exits; allow the program to complete execution normally.

Table 3: Error categories and corresponding feedback hints used in prompt augmentation.

E.1 Effect of English Translation

We first evaluate the role of bilingual translation. When the system relies solely on Bangla instructions without their English counterparts, comprehension drops significantly. The LLM often fails to parse algorithmic phrases and control keywords written in Bangla. As shown in Table 4, removing the translation stage reduces Pass@1 accuracy by

Main Prompt Template for Coder

```
{examples}
» Your Task
> Instruction
“‘python
def {function_call}:
"""{instruction}"""
"""Translated: {instruction_en}"""
"""{docstring}"""
“‘
Now complete the python code for the function ’{function_name}’ and add a
’main’ function with unit tests. You should use the ’check’ function for unit tests,
which is helpful for debugging. For example:
“‘python
def {function_call}:
# Your code

def check(test_id, test_val, expected):
assert test_val == expected, f”Test {test_id}: Expected {expected}, got
{test_val}”

def main():
{check_example}
# Add more unit tests
“‘
```

Figure 4: Main prompt template for the coder, which includes few-shot examples and the current task.

Failed Attempt Feedback Template

```
» Last failed code
> Response:
{last_response}
> Error:
{last_error}
> Suggested Fix:
{fix_instructions}
```

Figure 5: Template for providing feedback to the coder model after a failed execution attempt. This is appended to the main prompt during the self-refinement loop.

nearly 22 percent, confirming that current models still struggle to reason directly over Bangla-only text.

Setting	Pass@1 (%)
Full Model (Bangla + English)	95.5
Bangla Only	73.6

Table 4: Effect of English translation on Pass@1 accuracy (Lg Exaone Deep 32B, Dev Set).

E.2 Effect of Glossary-based Translation

We also analyze the impact of the controlled translation glossary used for mathematical and algorithmic terms. Without this glossary, the translation model often produces inconsistent or incorrect terminology, confusing the Coder during reasoning. As shown in Table 5, removing the glossary results in a notable performance drop of over 7 points, confirming that LLMs struggle to translate some Bangla words properly, leading to incorrect function generation.

Setting	Pass@1 (%)
With Glossary (Full Model)	95.5
Without Glossary	88.2

Table 5: Effect of using the controlled translation glossary.

E.3 Effect of Feedback Loop

Next, we disable the iterative self-refinement mechanism. Without execution feedback or re-prompting, the model cannot correct runtime or

System Prompt for Reviewer Model

You are a Python code reviewer and programming assistant.
The user will provide a function stub or implementation where the original docstring
is written in Bangla with a translated English version, along with unit test cases.
Your task is to:
1. Do not alter the given function signature.
2. Review the implementation for correctness, clarity, efficiency, and robustness.
3. Refactor or improve the implementation if needed, but the function signature must
remain identical.
4. Ensure the function works correctly not only for the provided tests but also for
hidden test cases and **corner cases** (e.g., empty inputs, boundary values,
invalid values, very large inputs).
5. Add a `main` function with unit tests that use the provided check function.
6. Include the given test cases and add additional edge/corner case tests that a
hidden evaluator might check.
7. Do not add explanations, comments, or extra text. Just return the code solution.

Important guidelines:
1. The function signature is already provided. Implement or refactor the function
as specified.
2. Include a **main function** (using `def main():`) that contains both the given
unit tests and extra corner/hidden-case tests you find necessary.
3. Do **not** call `main()` anywhere in your code. It will be executed externally.
4. Keep the code clean, correct, and as simple as possible while ensuring it passes
all tests, including edge and hidden cases.
5. Your response must be only one valid Python code block enclosed in triple backticks:
“‘python
`# your code here`
“‘

Figure 6: System prompt for the reviewer model.

Main Prompt Template for Reviewer

» Your Task
The following function is already implemented:
`"""{instruction}"""`
`"""Translated: {instruction_en}"""`
“‘python
`{existing_code}`
“‘

Figure 7: Main prompt template for the reviewer model.

logic errors, leading to a steep performance drop. Table 6 shows that accuracy declines by more than 25 percent, emphasizing that feedback-driven correction is vital for reliable synthesis.

Setting	Pass@1 (%)
Full Model	95.5
Without Feedback Loop	69.8

Table 6: Impact of feedback-driven refinement.

E.4 Effect of Reviewer LLM

To measure the Reviewers contribution, we bypass the second-stage review and directly execute the Coder output. Although the generated code remains mostly functional, it lacks stylistic polish and robustness on edge cases. Table 7 shows a moderate decline of about 5 percent, verifying that the Reviewer mainly improves coverage and reliability.

Setting	Pass@1 (%)
Full Model	95.5
Without Reviewer	90.4

Table 7: Effect of disabling the Reviewer LLM.

Few-shot Example Template

```
» Example {idx}:
> Instruction
“‘python
def {function_call}:
"""{instruction}"""
"""Translated: {instruction_en}"""
"""{docstring}"""
“‘
> Solution
“‘python
{solution}

def check(test_id, test_val, expected):
assert test_val == expected, f"Test {test_id}: Expected {expected}, got
{test_val}"

def main():
{test_main}
“‘
```

Figure 8: Template for formatting each of the k-nearest examples for retrieval-augmented generation.

System Prompt for Translator Model

Translate the following Bangla Python Code Instruction to English and only return the English translation. Do not change the example function and parameter names and only update the function parameter types and return variable types of Example function prototype to actual python syntax based on the provided unit test. Do not give the full code implementation. Just give the updated prototype.

Use the following glossary for translation: {glossary}

Unit Test: {test}

Figure 9: System prompt for the translator model.

E.5 Number of Feedback Iterations

We vary the maximum feedback iterations (M) to observe convergence behavior. As shown in Table 8, fewer iterations significantly reduce success rate since many tasks require multiple refinement cycles. Beyond five iterations, improvements saturate.

Max Iterations (M)	Pass@1 (%)
1	84.1
3	92.4
5	95.5
7	95.5

Table 8: Effect of limiting feedback iterations (M).

E.6 Effect of Retrieval Augmentation (RAG)

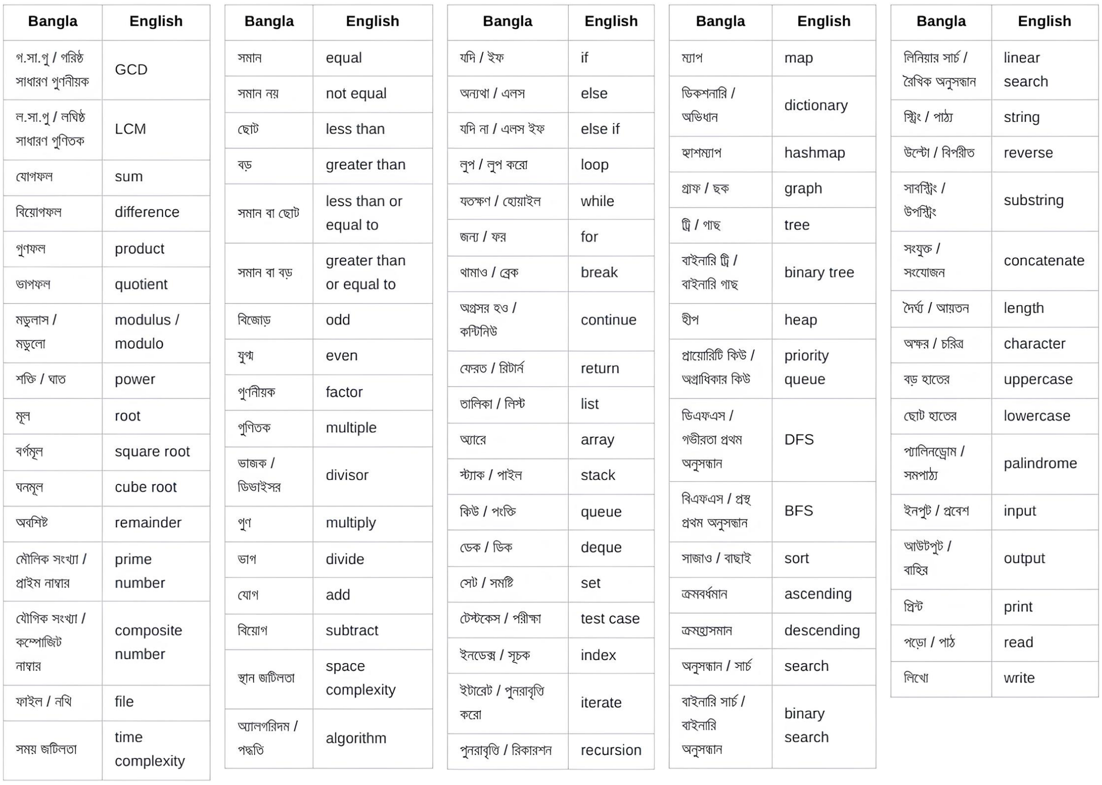
We compare our retrieval-augmented setup against a manually few-shot configuration. In the manual setup, the examples are fixed and not selected dynamically based on similarity, while the RAG variant retrieves the top-k relevant bilingual examples for each new task. As Table 9 shows, retrieval augmentation provides a small but consistent improvement of about 1.3 points, indicating that example relevance matters more than sheer quantity.

Setting	Pass@1 (%)
With RAG (Full Model)	95.5
Manual Few-shot (Fixed Examples)	94.2

Table 9: Comparison between manual few-shot and RAG-based prompting.

E.7 Number of Retrieved Examples (k)

Finally, we study the impact of the retrieval context size. As Table 10 shows, removing examples ($k = 0$) severely hampers the models grounding ability, dropping performance below 70%. Accuracy improves steadily up to $k = 5$, after which



Bangla	English
গ.সা.গু / গরিষ্ঠ সাধারণ গুণনীয়ক	GCD
ল.সা.গু / লঘিষ্ঠ সাধারণ গুণিতক	LCM
যোগফল	sum
বিয়োগফল	difference
গুণফল	product
ভাগফল	quotient
মডুলাস / মডুলো	modulus / modulo
শক্তি / ঘাত	power
মূল	root
বর্গমূল	square root
ঘনমূল	cube root
অবশিষ্ট	remainder
মৌলিক সংখ্যা / প্রাইম নাম্বার	prime number
যৌগিক সংখ্যা / কম্পোজিট নাম্বার	composite number
ফাইল / নথি	file
সময় জটিলতা	time complexity

Bangla	English
সমান	equal
সমান নয়	not equal
ছোট	less than
বড়	greater than
সমান বা ছোট	less than or equal to
সমান বা বড়	greater than or equal to
বিজোড়	odd
যুগ্ম	even
গুণনীয়ক	factor
গুণিতক	multiple
ভাজক / ডিভাইসর	divisor
গুণ	multiply
ভাগ	divide
যোগ	add
বিয়োগ	subtract
স্থান জটিলতা	space complexity
অ্যালগরিদম / পদ্ধতি	algorithm

Bangla	English
যদি / ইফ	if
অন্যথা / এলস	else
যদি না / এলস ইফ	else if
লুপ / লুপ করো	loop
যতক্ষণ / হোয়াইল	while
জন্য / ফর	for
থামাও / ব্রেক	break
অগ্রসর হও / কন্টিনিউ	continue
ফেরত / রিটার্ন	return
তালিকা / লিস্ট	list
অ্যারে	array
স্ট্যাক / পাইল	stack
কিউ / পংক্তি	queue
ডেক / ডিক	deque
সেট / সমষ্টি	set
টেস্টকেস / পরীক্ষা	test case
ইনডেক্স / সূচক	index
ইটারেট / পুনরাবৃত্তি করো	iterate
পুনরাবৃত্তি / রিকারশন	recursion

Bangla	English
ম্যাপ	map
ডিকশনারি / অভিধান	dictionary
হ্যাশম্যাপ	hashmap
গ্রাফ / ছক	graph
ট্রি / গাছ	tree
বাইনারি ট্রি / বাইনারি গাছ	binary tree
হীপ	heap
প্রায়োরিটি কিউ / অগ্রাধিকার কিউ	priority queue
ডিএফএস / গভীরতা প্রথম অনুসন্ধান	DFS
বিএফএস / প্রস্থ প্রথম অনুসন্ধান	BFS
সাজাও / বাছাই	sort
ক্রমবর্ধমান	ascending
ক্রমহ্রাসমান	descending
অনুসন্ধান / সার্চ	search
বাইনারি সার্চ / বাইনারি অনুসন্ধান	binary search

Bangla	English
লিনিয়ার সার্চ / রৈখিক অনুসন্ধান	linear search
স্ট্রিং / পাঠ্য	string
উল্টো / বিপরীত	reverse
সাবস্ট্রিং / উপস্ট্রিং	substring
সংযুক্ত / সংযোজন	concatenate
দৈর্ঘ্য / আয়তন	length
অক্ষর / চরিত্র	character
বড় হাতের	uppercase
ছোট হাতের	lowercase
প্যালিনড্রোম / সমপাঠ্য	palindrome
ইনপুট / প্রবেশ	input
আউটপুট / বাহির	output
প্রিন্ট	print
পড়ো / পাঠ	read
লিখো	write

Figure 10: Glossary for the translation prompt

marginal gains diminish due to context saturation.

Number of Examples (k)	Pass@1 (%)
0 (No Examples)	69.3
3	88.9
5 (Full)	95.5
7	94.7

Table 10: Effect of retrieved example count (k).

E.8 Comprehensive Summary

Table 11 consolidates all variants. The results confirm that English translation and the feedback loop contribute the largest performance boosts, while the glossary, reviewer, and RAG components further improve consistency, code quality, and generalization.

F Algorithm

Algorithm 1 shows the pseudocode of our pipeline.

G Failure Cases and Dataset Limitations

The dataset for Bangla-to-Python code generation was created by translating existing English datasets MBPP (Mostly Basic Python Problems) and HumanEval into Bangla using machine translation. While this approach enables rapid dataset construction, it introduces several limitations that affect both dataset quality and model performance.

G.1 Semantic and Syntactic Translation Errors

Machine translation occasionally produces Bangla sentences that are grammatically incorrect or semantically ambiguous. Such translations may hinder a models ability to correctly interpret the input and generate the intended Python code. For example:

- The English adjective “even” was translated as এমনকি instead of the more contextually accurate জোড় in cases where *even* refers to parity in numbers. This leads to semantic confusion and misinterpretation of the question context.

G.2 Incorrect or Misleading Terminology for Programming Concepts

Programming terms often lack direct equivalents in Bangla. Machine translation systems attempt to

Configuration	Translation	Glossary	Feedback Loop	Reviewer	RAG	Pass@1 (%)
Full BanglaForge Pipeline	Yes	Yes	Yes	Yes	Yes	**95.5**
Without Translation	No	Yes	Yes	Yes	Yes	73.6
Without Glossary	Yes	No	Yes	Yes	Yes	88.2
Without Feedback Loop	Yes	Yes	No	Yes	Yes	69.8
Without Reviewer	Yes	Yes	Yes	No	Yes	90.4
Manual Few-shot (No RAG)	Yes	Yes	Yes	Yes	No	94.2
Fewer Iterations (M = 1)	Yes	Yes	Yes	Yes	Yes	84.1
Fewer Examples (k = 3)	Yes	Yes	Yes	Yes	Yes	88.9
No Examples (k = 0)	Yes	Yes	Yes	Yes	Yes	69.3

Table 11: Comprehensive ablation results on the BLP-2025 development set using Lg Exaone Deep 32B.

Algorithm 1 Algorithm of BanglaForge

```
Input: BanglaInstruction $P_b$, PublicUnitTests $T$
Output: ExecutableCode
$M \leftarrow$ maximum retry limit
$attempt \leftarrow 0$
EnglishInstruction, $P_e \leftarrow$ TranslatorLLM.translate($P_b$)
Examples, $E \leftarrow$ Database.retrieveExamples($P_b$, $P_e$)
PromptCoder $\leftarrow$ constructPrompt($P_b$, $P_e$, $T$, $E$)
while $attempt < M$ do
    $attempt \leftarrow attempt + 1$
    ($c$, $T_c$) $\leftarrow$ CoderLLM.generate(PromptCoder)
    PromptReviewer $\leftarrow$ constructReviewPrompt($c$, $T_c$)
    ($c_r$,$T_r$) $\leftarrow$ ReviewerLLM.refine(PromptReviewer)
    Result $\leftarrow$ executeCode($c_r$, $T \cup T_c \cup T_r$)
    if Result.allTestsPassed then
        return $c_r$
    else
        Feedback $\leftarrow$ generateFeedback(Result.errors)
        PromptCoder $\leftarrow$ updatePromptWithFeedback(PromptCoder, Feedback)
    end if
end while
```

generate literal translations, but these often fail to capture technical meaning. For example:

- The English term *“Map”* was translated to মানচিত্র (meaning a geographic map in Bangla), instead of referring to *Map* as in a data structure such as HashMap or dictionary. This causes ambiguity, making it challenging for both humans and models to interpret correctly.
- Similarly, terms like *stack*, *queue*, *hashmap*, or *dictionary* may be incorrectly translated, or not translated at all, resulting in inconsistent terminology across the dataset.

G.3 Loss of Context or Intent

Machine translation may fail to preserve the precise context or intent of the original English instructions. Programming problems often rely on subtle nuances, and even small changes in wording can alter the meaning of a problem. This issue is exacerbated when the translated text uses uncommon or unnatural phrasing, reducing clarity for model training.

G.4 Lack of Standardized Technical Vocabulary

Bangla currently lacks standardized technical vocabulary for many programming concepts, leading to inconsistent translations. In some cases, the same English term is translated differently across dataset entries. This inconsistency makes it difficult for a model to reliably learn the intended mapping from Bangla instructions to Python code.

G.5 Impact on Model Performance

These translation-related issues contribute to notable failure cases in Bangla-to-Python code generation. Models trained on such data may misinterpret problem statements, produce incorrect code, or fail to generalize to unseen examples. Addressing these limitations would require:

- Careful human curation of translations for correctness and consistency.
- Development of a standardized Bangla programming lexicon.
- Use of bilingual glossaries to retain original technical terms where necessary.

BRACU_CL at BLP-2025 Task 2: CodeMist: A Transformer-Based Framework for Bangla Instruction-to-Code Generation

Md. Fahmid-Ul-Alam Juboraj[1], **Soumik Deb Niloy**[1], **Mahbub E Sobhani**[1,2]
Farig Yousuf Sadeque[1,*]
[1]BRAC University [2]United International University
{md.fahmid.ul.alam.juboraj, soumik.deb.niloy}@g.bracu.ac.bd
msobhani2410011@mscse.uiu.ac.bd, farig.sadeque@bracu.ac.bd
∗ denotes corresponding author

Abstract

We propose CodeMist, a hybrid framework for Bangla-to-Python code generation, focusing on enhancing code accuracy through a two-stage pipeline of generation and debugging. In the development phase, standalone models such as `TigerLLM` and `StarCoder` achieved low accuracies of 27% and 24%, respectively, while advanced models like `Gemini-1.5-flash` and `Gemma` reached 60% and 64%. Pairing `Gemma` with the `GPT-OSS` debugger resulted in a substantial improvement to 99.75%, emphasizing the importance of a dedicated debugging stage. In the test phase on unseen data, `GPT-OSS` alone achieved 67%, which increased to 71% with self-debugging. The highest performance of 84% was achieved by combining `Gemini-2.5-flash` as the generator with `GPT-OSS` for debugging. These results demonstrate that integrating a strong generative model with an effective debugging component produces superior and robust code generation outcomes, outperforming existing approaches such as `TigerLLM`. The full implementation of the framework is publicly available at `https://github.com/fahmid-juboraj/Code_generation`.

1 Introduction

Automated code generation has witnessed rapid advancement with the emergence of Large Language Models (LLMs) such as *CodeT5* (Wang et al., 2021) and *CodeGen* (Nijkamp et al., 2022), which achieve over 90% accuracy on benchmarks like *HumanEval* (Raihan et al., 2025a). These models translate natural language instructions into executable code, substantially enhancing developer productivity and software development efficiency.

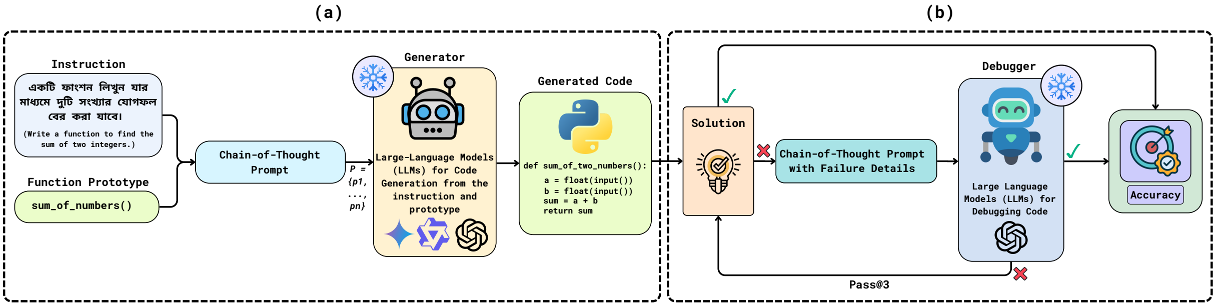
Despite these successes, most LLMs are trained primarily on English-based instructions, creating a linguistic bias in code generation research (Raihan et al., 2025c). This English-centric training restricts accessibility for non-English-speaking developers and undermines the inclusivity of AI-assisted programming tools. While languages such as Chinese and Japanese have begun to receive research attention, Bangla, the fifth most spoken language globally with over 300 million speakers, remains largely unexplored in this domain. Existing multilingual models struggle to generalize Bangla semantics to programming constructs due to the lack of high-quality Bangla–Python parallel datasets and the absence of specialized benchmarks for evaluation.

To address these limitations, this study introduces CodeMist, a hybrid Bangla-to-Python code generation framework that integrates generative modeling with automated debugging which employs the Gemini API to generate Python code from Bangla instructions and subsequently refines it through GPT-OSS, a locally fine-tuned GPT-based model capable of detecting and correcting syntax and logical errors. This dual-stage framework not only enhances the accuracy and correctness of generated code but also demonstrates the effectiveness of coupling code generation with an adaptive debugging mechanism for non-English programming tasks. The contribution lies in expanding code generation research beyond English, offering a foundational step toward linguistically inclusive AI programming systems.

2 Related Work

BanglaBERT (Bhattacharjee et al., 2022) pioneered Bangla-specific language modeling and introduced the BLEU benchmark for NLU evaluation. Subsequent models such as *TigerLLM* (Raihan et al., 2025d) and *TituLLMs* (Ahmed et al., 2025) improved reproducibility, coverage, and transliteration handling. Community-driven efforts like the *BanglaLLM* project (BanglaLLM Organization, 2024) released open-source models such as *Bangla-LLaMA-13B* (Zehady and the

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 656–662
December 23, 2025 ©2025 Association for Computational Linguistics


(a)
Instruction
Function Prototype
sum_of_numbers()
Chain-of-Thought Prompt
P = {p1, ..., pn}
Generator
Large-Language Models (LLMs) for Code Generation from the instruction and prototype
Generated Code
(b)
Solution
✓
✗
Chain-of-Thought Prompt with Failure Details
Debugger
Large Language Models (LLMs) for Debugging Code
✓
✗
Pass@3
Accuracy

একটি ফাংশন লিখুন যার মাধ্যমে দুটি সংখ্যার যোগফল বের করা যাবে।

(Write a function to find the sum of two integers.)

```
def sum_of_two_numbers():
    a = float(input())
    b = float(input())
    sum = a + b
    return sum
```

Figure 1: **(a)** A Chain-of-Thought prompt combines a natural-language instruction in Bangla with a function prototype, which is provided to the Generator to produce candidate implementations. **(b)** A Debugger constructs a Chain-of-Thought prompt with failure details, and debugging LLMs are engaged to iteratively repair and re-evaluate the generated code, where ✓ denotes correct code and ✗ denotes faulty code.

BanglaLLM Team, 2024), enabling wider experimentation. Parameter-efficient fine-tuning approaches such as *LoRA* (Hu et al., 2021) have been applied to resource-constrained models like *Gemma 2B* (Dasgupta, 2024) and *Bangla-LLaMA* (Saiful, 2023), demonstrating scalable adaptation methods.

The BLEU benchmark (Bhattacharjee et al., 2022) evaluates core Bangla NLU tasks, including sentence classification and sequence labeling. Later datasets introduced task-specific challenges such as tense classification in *BanglaTense* (Rahman et al., 2025) and sentiment analysis in *BnSentMix* (Data Analytics Research Group, 2024). Despite these advances, standardized benchmarks remain limited (Khan et al., 2025), restricting cross-model comparison. Community repositories like (Bangla NLP Community, 2024) consolidate datasets, yet coverage of generative tasks, especially code generation, is insufficient. *mHumanEval* addresses this gap with a massively multilingual benchmark, featuring prompts in 204 languages and solutions in 25 programming languages to evaluate cross-lingual coding performance.

While LLMs achieve strong performance in English code generation (Chen et al., 2021; Nijkamp et al., 2022), Bangla instruction-based code generation remains largely unexplored. Most Bangla LLMs target natural language understanding rather than code synthesis. Only a few, such as *TigerCoder* (Raihan et al., 2025b), focus on code generation, but they are limited by small datasets, lack of standardized evaluation, and minimal error-correction mechanisms. The absence of large-scale Bangla–Python parallel corpora and the challenge of mapping Bangla semantics to programming logic make reliable code generation non-trivial. Our work addresses these limitations by curating a comprehensive Bangla-to-Python dataset and developing a two-phase hybrid framework combining code generation with iterative debugging using models like *GPT-OSS*, achieving higher accuracy and robust synthesis on both development and unseen test sets.

3 Methodology

In this section, we outline the proposed pipeline for generating Python code from Bangla instructions, as shown in Figure 1.

3.1 Problem Formulation

We are studying the task of generating code from instructions provided in the Bangla language. Each problem consists of a Bangla instruction set, denoted as $INS = \{ins_1, ins_2, \dots, ins_n\}$, paired with a corresponding function prototype set, $FP = \{fp_1, fp_2, \dots, fp_n\}$ and explicit CoT prompt $CP = \{cp_1, cp_2, \dots, cp_n\}$. The objective is to generate the correct Python code, represented as $\hat{Y}$, for each pair (ins_i, fp_i). This framework enables us to compare different modeling approaches: code generator models that directly map from (ins_i, fp_i, cp_i) to $\hat{Y}$ and code debugger models that iteratively refine the generated code. Throughout the execution of our CodeMist pipeline, the parameters for all models were kept frozen. The entire procedure can mathematically be abbreviated as follows:

$$\hat{Y} = LM_{\text{dbg}}([cp,\, LM_{\text{gen}}(ins, fp, cp)]) \quad (1)$$

3.2 Motivation

Automated code generation from Bangla instructions often faces errors due to linguistic ambigu-

Generation	Debugging	Accuracy(%)
StarCoder	None	24.00
TigerLLM	None	27.00
Qwen-1.5B	None	27.00
TigerCoder	None	33.00
TigerCoder	GPT-OSS	53.00
Gemini-1.5-flash	None	60.00
Gemma	None	64.00
Gemma	**GPT-OSS**	**99.75**

Table 1: Performance of Code Generation and Debugging (Development Phase)

ity and model limitations. A pipeline that integrates code generation with recurring debugging enhances reliability while lowering manual effort. This approach enhances the accuracy and usability of generated programs, facilitating effective Bangla-to-code translation and evaluation.

3.3 CodeMist

In this section, we provide the details of our code generator-debugger pipeline.

3.3.1 Code Generator

The process begins with a CSV file containing multiple columns such as `instruction`, `id`, and other contextual features. Each instruction is first processed by a Prompt Design module, which transforms the input text into a structured prompt optimized for the Gemini API. This module may incorporate additional contextual information from the CSV, such as example inputs/outputs or constraints, to reduce ambiguity and improve model comprehension. The quality and clarity of these prompts are crucial, as they directly influence the correctness and functionality of the generated code. The prepared prompt is then sent to the Gemini API, which returns the generated code through its response interface. All outputs are stored in a JSON file containing each `id` and its corresponding response. The correctness of the generated code is subsequently evaluated using a Python script, `Scoring.py`. This evaluation informs both performance metrics and areas where debugging may be necessary.

3.3.2 Code Debugger

For cases where the generated code is incorrect or suboptimal, **CodeMist** activates the Code Debugger pipeline. This phase begins by logging errors and linking them with the corresponding instructions and failure reasons in an updated CSV file. A local model interface is initialized by downloading and configuring an appropriate model (e.g., GPT-OSS:20B) along with a system prompt that combines the original instruction, contextual information, and the identified failure reason. This enriched prompt guides the local model to produce corrected versions of the code. By explicitly including both the original instruction and failure context, the debugger can resolve syntax, logical, and runtime errors more effectively. Each erroneous case is reprocessed to generate improved outputs, which replace the previous responses and are saved in a new submission JSON file following the same schema (`id`, `response`). The corrected codes are subsequently evaluated using `Scoring_V2.py` to enable comparative performance analysis between the initial Gemini-generated results and the refined open-source LLM-based solutions.

Generation	Debugging	Accuracy(%)
GPT-OSS	None	67.00
GPT-OSS	GPT-OSS	71.00
Gemini-2.5-flash	**GPT-OSS**	**84.00**

Table 2: Performance of Code Generation and Debugging (Test Phase)

4 Experimental Analysis

4.1 Dataset

We utilize two datasets provided by the BLP Workshop (Raihan et al., 2025c) for training, validation, and evaluation of Bangla-to-Python code generation models. Both datasets contain Bengali problem descriptions paired with corresponding Python test cases but do not include ground-truth solutions. The `dev_v2.csv` dataset comprises 400 tasks and is used for model development and hyperparameter tuning, while the `test_v1.csv` dataset includes 500 tasks and serves as the held-out evaluation set. Each instance contains three fields—`id`, `instruction`, and `test_list`—where `instruction` provides the problem statement in Bengali and `test_list` defines the functional requirements in Python. Together, these datasets enable systematic assessment of code generation models' ability to generalize from natural language to executable programs.

4.2 Performance Evaluation

During the development phase, predictions are evaluated using a static checker that executes all assertion-based test cases. The metric is defined as:

$$\text{Pass@1}_{\text{dev}} = \frac{N_{\text{PASS}}}{N_{\text{TOTAL}}} \times 100\% \tag{2}$$

where N_{PASS} denotes the number of tasks whose generated code passes all test cases, and N_{TOTAL} is the total number of evaluated tasks.

For the final evaluation phase, a runtime-based evaluator with timeout and exception handling computes the functional correctness as:

$$\text{Pass@1}_{\text{final}} = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{3}$$

where N_{correct} represents the number of tasks whose generated programs pass all functional tests under execution, and N_{total} is the total number of tasks evaluated.

4.3 Experimental Results

4.3.1 Quantitative Results

During the development phase, several code generation models were evaluated independently and in combination with debugging support. As shown in Table 1, *TigerCoder* (Raihan et al., 2025b) achieved an accuracy of 33% without debugging, which improved to 53% when paired with the *GPT-OSS* (Community and Contributors, 2024) debugger. This illustrates the benefit of an explicit debugging phase in correcting syntax and logical errors in the generated code. Similarly, other standalone generation models such as `StarCoder` (Li et al., 2023) and *Qwen-1.5B* (Team, 2024c) achieved relatively low accuracies of 24% and 27%, respectively, indicating limited capability in producing fully functional code from Bangla instructions without assistance. In contrast, more advanced models like *Gemini-1.5-flash* (Team, 2024a) and *Gemma* (Team, 2024b) achieved higher accuracies of 60% and 64%, demonstrating improved contextual understanding and code synthesis. The most notable improvement was observed when Gemma was combined with the GPT-OSS debugger, achieving a remarkable accuracy of 99.75%. This underscores the significant role of the debugging component in refining model outputs. Overall, these findings confirm that integrating a powerful generator with a specialized debugger can dramatically enhance performance.

Subsequently, in the test phase, models were evaluated on unseen prompts to measure generalization. As shown in Table 2, *GPT-OSS* alone achieved a baseline accuracy of 67%, which increased to 71% with self-debugging (GPT-OSS + GPT-OSS). The combination of *Gemini-2.5-flash* (Team, 2024a) as the generator and *GPT-OSS* as the debugger achieved the best performance of 84%, highlighting the robustness of cross-model debugging.

4.3.2 Qualitative Analysis of Failures

To categorize failure types, we clustered failure descriptions based on their top keywords and manually interpreted the resulting groups.

Logical Failures. Keywords such as `testlist`, `comma`, and `syntax` indicate *structural or parsing errors* caused by formatting mistakes or missing elements.

Analytical Failures. Terms like `character`, `string`, and `execution` reflect *runtime or reasoning errors*, requiring understanding of expected program behavior.

Mathematical Failures. Keywords including `test`, `assertion`, and `exception` correspond to *validation or computation errors* such as failed assertions or unmet conditions.

5 Conclusion

This study demonstrates the effectiveness of a hybrid LLM-based pipeline for Bangla programming instruction understanding and Python code generation. By leveraging the Gemini API for initial synthesis and a local Ollama model for targeted correction, our approach overcomes language-specific limitations and achieves substantial performance gains. The iterative design ensures both scalability and adaptability, enabling error tracking, prompt refinement, and systematic evaluation. The improved accuracy underscores the value of integrating cloud-based and local models to enhance multilingual code generation. Future work will explore fine-tuning domain-specific Bangla LLMs, expanding datasets with richer semantic coverage, and applying reinforcement-based evaluation to further optimize generation quality.

6 Limitations

While our study provides valuable insights into Bangla to Python generation and the benefits of a generator–debugger pipeline, it has several limitations. First, all models were evaluated in a Chain-of-Thought (Wei et al., 2022) setting without fine-tuning, limiting our understanding of how training or domain adaptation might affect results. Second, we relied on fixed prompts and debugging strategies, without exploring adaptive or iterative prompting that could refine reasoning. Third, our evaluation used a limited dataset, restricting conclusions about generalization and robustness. Finally, our assessment focused solely on automated correctness metrics, without human or qualitative analyses to capture broader usability and failure patterns.

References

Khandaker Tahmid Ahmed, Mahir Rahman, Taufiq Islam, and Sabbir Khan. 2025. Titullms: A family of bangla llms with comprehensive benchmarking. *arXiv preprint arXiv:2502.11187*.

Bangla NLP Community. 2024. Awesome datasets for bangla language computing. GitHub Repository.

BanglaLLM Organization. 2024. Banglallm: Bangla large language model. Hugging Face Model Hub.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Uddin Ahmad, Yuan Li, Yong-Bin Kang, 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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

OpenAI Community and Contributors. 2024. Gpt-oss: Open-source gpt-like models for code understanding. https://github.com/openai. Used as Debugging Model.

Abhishek Dasgupta. 2024. Building a bangla llm by finetuning gemma 2b llm using low-rank-adaptation. *Medium Blog Post*.

Data Analytics Research Group. 2024. Bnsentmix: A large-scale bengali-english code-mixed sentiment analysis dataset.

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 preprint arXiv:2106.09685*.

Sabbir Ahmed Khan, Md Arafatur Rahman, Md Saiful Islam, and Kazi Sakib Ahmed. 2025. Evaluating llms' multilingual capabilities for bengali. *arXiv preprint arXiv:2507.23248*.

Raymond Li, Loubna Ben Allal, Peter Izsak, and et al. 2023. Starcoder: A large language model for code generation. https://huggingface.co/bigcode/starcoder. BigCode Project.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. In *International Conference on Learning Representations*.

Md Mahfuzur Rahman, Md Saiful Islam, Kazi Sakib Ahmed, and Sabbir Khan. 2025. Banglatense: A large-scale dataset of bangla sentences categorized by tense. *Data in Brief*, 53:110132.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11432–11461, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025b. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025c. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics (ACL).

Nishat Raihan, Joanna CS Santos, and Marcos Zampieri. 2025d. Tigerllm - a family of bangla large language models. *arXiv preprint arXiv:2503.10995*.

Mohammed Saiful. 2023. Bangla llama: Finetune bangla llama model using lora approach. GitHub Repository.

Google DeepMind Team. 2024a. Gemini 1.5: Unlocking multimodal capabilities in long contexts. https://deepmind.google/technologies/gemini/. Accessed: 2025-09-27.

Google DeepMind Team. 2024b. Gemma: Lightweight open models from google deepmind. https://ai.google.dev/gemma. Accessed: 2025-09-27.

Qwen Team. 2024c. Qwen1.5: A comprehensive multilingual large language model. https://huggingface.co/Qwen. Accessed: 2025-09-27.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Abdullah Khan Zehady and the BanglaLLM Team. 2024. Bangla llama 13b instruct v0.1. Hugging Face model card, `https://huggingface.co/BanglaLLM/bangla-llama-13b-instruct-v0.1`. Open-source Bangla instruction-tuned large language model (13B parameters).

A Appendix

A.1 Dataset Sample

This table provides a few representative samples from our dataset. Each row shows the sample ID, the corresponding task instruction in Bangla, and the Python assertions used to verify the expected output.

Table 3: Sample instances from the Dataset

ID	Instruction	Test List (Sample)
1	একটি ফাংশন লিখুন যা দিয়ে দেওয়া জোড়া থেকে তৈরী করা যায় সর্বাধিক শৃঙ্খল।	[’assert max_chain_length([Pair(5, 24), Pair(15, 25)]) == 3’]
2	একটি প্রদত্ত স্ট্রিং-এ প্রথম পুনরাবৃত্ত অক্ষর নির্ণয় করুন।	[’assert first_repeated_char(”abcabc”) == ”a”’]
3	একটি ফাংশন লিখুন যা n এর চেয়ে ছোট বা সমান একটি লুডিক সংখ্যা বের করবে।	[’assert get_ludic(10) == [1, 2, 3, 5, 7]’]
4	একটি প্রদত্ত স্ট্রিং এর শব্দগুলোকে বিপরীত করার প্রোগ্রাম লিখুন।	[’assert reverse_words(”python program”) == ”program python”’]
5	প্রদত্ত পূর্ণসংখ্যাটি একটি মৌলিক সংখ্যা কিনা তা যাচাই করুন।	[’assert prime_num(13) == True’]

A.2 Sample of Generated Failed With Details

This table shows the first five samples from `failed_with_details_test.csv`. Each row lists the sample ID, the type of failure encountered during testing, the task instruction in Bangla, and the corresponding Python test list that caused the failure.

Table 4: Samples from `failed_with_details_test.csv`

ID	Failures	Instruction	Test List
3	[X][X][X][X] Failed to parse test_list: invalid syntax near “,”.	একটি ফাংশন লিখুন যা একটি অভিধানে সবচেয়ে সাধারণ মানটি খুঁজে বের করবে।	[’assert count_common ([’red’,’green’,’black’], [’green’,’white’,’red’]) == 2’]
5	[X][X][X][X] Failed to parse test_list: invalid syntax near “]”.	একটি স্ট্রিংকে ছোট অক্ষরে বিভক্ত করার জন্য একটি ফাংশন লিখুন।	[’assert split_lowerstring (”AbCd”) == [’bC’,’d’]’]
6	[X][X][X][X] Failed to parse test_list: invalid syntax near “_”.	একটি ফাংশন লিখুন যাতে ছোট অক্ষরের ক্রম খুঁজে পাওয়া যায়।	[’assert text_lowercase_underscore (”aab_cbbbc”) == True’]
8	[X][X][X][X] Failed to parse test_list: invalid syntax near “(”.	প্রথম স্ট্রিং থেকে দ্বিতীয় স্ট্রিংয়ে উপস্থিত অক্ষরগুলি মুছে দিন।	[’assert str_to_list (”probasscurve”, ”pros”) == [”b”,”a”,”c”,”u”,”v”,”e”]’]
15	[X] Error in function definition: unexpected character.	একটি প্রদত্ত অ্যারেতে পুনরাবৃত্তি না হওয়া উপাদানগুলির যোগফল নির্ণয় করুন।	[’assert find_Product ([1,1,2,3],4) == 6’]

Code_Gen at BLP-2025 Task 2: BanglaCode: A Crosslingual Benchmark for Code Generation with Translation and Assertion Strategies

Abhishek Agarwala[1]* **Shifat Islam**[1]* **Emon Ghosh**[2]*
Department of Computer Science
[1]Bangladesh University of Engineering and Technology
[2]Ahsanullah University of Science and Technology
{abhishek.agarwal0395, shifat.islam.buet, emonghosh005}@gmail.com

Abstract

Large Language Models (LLMs) have shown great code-generation capabilities, but their performance in low-resource languages like Bangla is largely unexplored. We participated in BLP-2025 Task 2: Code Generation in Bangla, where we built a pipeline to interpret and execute Bangla instructions using GPT-5. Extensive experiments were conducted with proprietary (GPT-4o Mini, GPT-5 Mini, GPT-5) and open-source (LLaMA 3-8B, TigerLLM-1B-it) models under translation and assertion settings. Results show that GPT-5, with translation and assertion, scored **83.8%**, outperformed all baselines, while open-source models lagged due to limited Bangla adaptation. Assertion-based prompting always improved syntactic correctness, and fine-tuning reduced hallucinations across open-source models. We ranked **7th** on the official leaderboard with an approach which is competitive and generalizable. Overall, our results show that translation quality, data normalization, and prompt design are key components of low-resource code generation. Furthermore, the proposed BanglaCode benchmark and preprocessing architecture provide a basis for further multilingual code-generation research.

1 Introduction

Recent advancements in Large Language Models (LLMs) have made it possible to generate code from natural language descriptions and changed the landscape of software development (Brown et al., 2020). Models such as GPT-3.5 and GPT-4 have demonstrated high performance for a wide range of programming tasks(Coello et al., 2024). But the performance of these models is limited in low-resource languages like Bangla, spoken by more than 242 million people[1]. The gap in Bangla NLP is mainly due to a lack of high-quality, language-specific training datasets and models(Zehady et al., 2024), which limits the performance of existing multilingual LLMs when applied to Bangla.

This paper addresses these gaps by evaluating and improving Bangla code generation through various strategies like preprocessing, translation, prompting and fine-tuning. Specifically, this paper focuses on understanding how different approaches impact the performance of LLMs when generating code in Bangla.

Our contributions are:

- **Preprocessing**: Applied noise reduction techniques like removing special characters and repeated words to refine Bangla code instructions and improve model input and code generation accuracy.
- **Translation**: Translated Bangla code instructions to English using Google Translate, facebook/nllb-200-distilled-600M, and GPT-5 and checked if translation-induced semantic loss affects code generation.
- **Prompting and Fine-Tuning**: We introduce assertion-based prompting by varying the number of assertions in input prompts and fine-tune pre-trained models on a curated Bangla code instruction dataset and optimize for Bangla-specific tasks.
- **Benchmark Comparison**: Compare open-source models (LLaMA 3, and TigerLLM-1B-it) and proprietary models (GPT-4o Mini, GPT-5 Mini, and GPT-5) on BanglaCode benchmark and see the effect of translation, preprocessing and assertion on code generation.

Through these, we want to create a new benchmark for Bangla code generation and evaluate transla-

*Equal contribution.

[1] https://en.wikipedia.org/wiki/Bengali_language#cite_note-e28/ben/Bengali-1

Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025), pages 663–668
December 23, 2025 ©2025 Association for Computational Linguistics

tion, preprocessing and prompting strategies. By providing an open-source benchmark and new preprocessing and prompt optimization techniques, we want to contribute to future research in Bangla and other low-resource languages. This will not only advance the understanding of code generation in low-resource languages but also make LLMs more effective for code generation in different linguistic contexts.

Both our code and training corpus are publicly available in the GitHub repository[2].

2 Related Work

Large Language Models (LLMs) have made a lot of progress in code generation with models like Codex (Chen et al., 2021), StarCoder (Li et al., 2023), and Code LLaMA (Roziere et al., 2023) doing well on benchmarks like HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). However, these benchmarks and models are overwhelmingly English-centric and not applicable in multilingual or low-resource settings. In order to assess LLMs on code creation from natural language prompts in 204 languages, including Bangla, mHumanEval (Raihan et al., 2024) established a multilingual benchmark. Their findings demonstrate that even the most advanced multilingual LLMs, such as GPT-3.5 and GPT-4, perform much worse on low-resource languages, which is known as the language gap (Coello et al., 2024).

Bangla is severely underrepresented in code-related NLP research despite being the 5th most spoken language in the world. Existing multilingual models like LLaMA 3 (Grattafiori et al., 2024), BLOOM (Workshop et al., 2022), and AYA (Üstün et al., 2024) have minimal Bangla content in their training data and perform poorly when prompted in Bangla (Raihan et al., 2024).TigerCoder (Raihan et al., 2025a) is the first dedicated Bangla code-generation LLM. It introduces three new instruction datasets (self-instructed, synthetic, and translated), and a Bangla version of the MBPP benchmark (MBPP-Bangla), covering five programming languages. TigerCoder models gain 11–18% absolute over general-purpose Bangla LLMs and multilingual baselines, proving the importance of domain-adapted training for code generation in low-resource languages.

3 Task Description

Code generation is becoming more important in natural language processing (NLP), and this work aims to extend it to the Bangla language by addressing the challenges and proposing solutions. Task 2 of the Bangla Language Processing (BLP) (Raihan et al., 2025b) focuses on code generation in Bangla by asking participants to develop systems that can translate Bangla prompts into Python scripts that can pass hidden unit tests. Each dataset instance contains a Bangla instruction describing a programming problem, a reference Python implementation included in the trial dataset, and a set of unit tests. By comparing different approaches in terms of robustness and applicability, this task not only shows the feasibility of Bangla code generation but also its potential for broader NLP applications, so that Bangla can be integrated into everyday computational tasks and make the technological landscape more inclusive and connected.

3.1 Dataset Description

The dataset (Raihan et al., 2024, 2025a) used in this study was from the BLP Workshop Task 2 on Code Generation in Bangla. Table 1 shows the sample of data from the given dataset. It was divided into three parts: trial set, dev set, and test set, with 74, 400, and 500 samples, respectively. The trial.csv was designed to fine-tune the model to capture the Bangla language properties and nuances.

id	instruction	test_list

4	একটি ত্রিভুজাকার প্রিজমের আয়তন খুঁজে বের করার জন্য একটি পাইথন ফাংশন লিখুন।
	Example:
	def find_Volume(l,b,h):
	    # your code
	    return l
		['assert find_Volume(10,8,6) == 240']

8	প্রথম স্ট্রিং থেকে দ্বিতীয় স্ট্রিংয়ে উপস্থিত অক্ষরগুলি সরিয়ে ফেলার জন্য একটি ফাংশন লিখুন।
	Example:
	def str_to_list(string):
	    # your code
	    return string
		['assert str_to_list (\”probasscurve\”, \”pros\”) == \'bacuve\']

Table 1: Bangla Instructions with Python Code and Test Cases of Sample Data

Each sample in the dataset had four fields: id, instruction, response, and test_list. The id column was a unique identifier for each sample. The instruction column was the main task in Bangla,

[2] https://github.com/ShifatIslam/BLP-Task-2

which guided the code generation. The response column was only in the trial set and contained the target code solutions and was the ground truth for fine-tuning. The test_list column contained input–output pairs and was used to verify the correctness of the generated code. The dev.csv and test_full.csv did not have the response column. They only had id, instruction, and test_list. These subsets were only for code generation, where the task was to generate programs that satisfy the assertions in test_list according to the Bangla instructions.

4 Experimental Setup

The overall methodology of our study is shown in Figure 1. We prepare and translate the dataset, then extract and align assertions to make the instructions clearer. Then we refine the inputs through prompt tuning. Finally, we train and evaluate different models on BanglaCode to see how translation, assertions, and prompting affect code generation.

4.1 Dataset Preparation and Translation

The dataset had 3 fields: instruction, id, text_list. Upon closer inspection, the instruction field had discrepancies and irregularities among the texts. These included erroneous texts, repetition of the same words, and unnecessary texts. Direct translation was not possible. A preprocessing pipeline was built to normalize the data and make the text consistent.

After the dataset was cleaned, translation was done. Several open-source models were tested: Google Translate and Facebook/nllb-200-distilled-600M. These models failed miserably with semantic errors and synonym inconsistencies. GPT-5 was chosen for the final translation step because it was the best at fluency and semantic accuracy.

4.2 Assertion Extraction and Alignment

Translation alone was not enough to improve the performance of the downstream code generation. To strengthen the dataset, assertions were extracted and appended to the translated instructions. The raw assertions were problematic: mismatched functions and syntax errors, extraneous commas, irregular use of brackets, and misplaced quotation marks. An algorithm was built to extract, clean and align the assertions with the instructions. The assertions provided clearer guidance for code generation and overall robustness.

4.3 Prompt Optimization

Prompt design was also experimented with. Short and direct prompts outperformed longer prompts with redundant or incorrect examples. This was true across all models, including GPT-5. Prompt engineering is key to code generation performance.

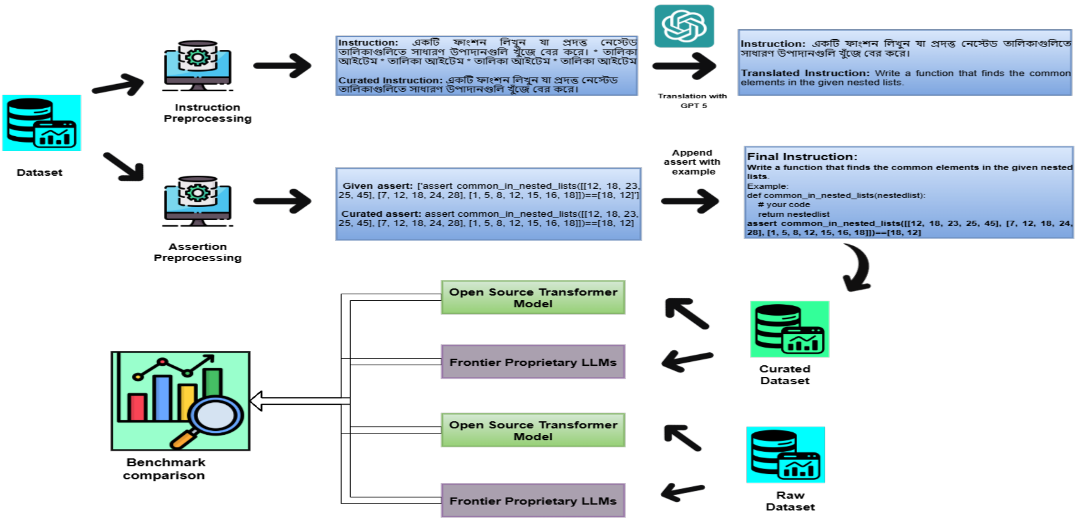
4.4 Model Training and Evaluation

Multiple models were tried for the code generation task. Open-source models like LLaMA-3 and TigerLLM-9B were not accurate due to hardware constraints, hallucinations, and their inability to perform well due to their pretraining and size. Fine-tuning was attempted on a curated dataset (trial.csv) with 74 instruction–response pairs. Although this improved consistency, the small sample size limited the performance gain. Proprietary models via API: GPT-4-O Mini, GPT-5 Mini, and GPT-5 were used. GPT-5 with translated texts and appended assertions scored 83.4% and ranked 7th.

5 Result Analysis and Findings

From table 2, we can see the accuracy of different models. As expected, the proprietary models performed significantly better than the open-source models. This comes as no surprise, as the paid models are state-of-the-art models with better reasoning and size. However, there are some caveats. The quality of performance boosted significantly when we switched from without translation to translation with assertions. However, even from models such as GPT-5 the increase in accuracy is significant, which goes from 60.69% to 83.80%. We can see this trend in all the models, including open-source models. This jump for GPT-5 shows that even though LLM models are taking over the world, there is significant improvement needed for languages such as Bangla.

We finetuned the open-source models with the trial dataset to reduce hallucinations, as accuracy remained low with occasional hallucinations despite a strong system prompt, showing the importance of model size for effective code generation. Additionally, we observed that prompt tuning plays a critical role in boosting accuracy. This was found by experimenting with small,


Dataset
Instruction Preprocessing
Assertion Preprocessing
Translation with GPT 5
Append assert with example
Curated Dataset
Raw Dataset
Open Source Transformer Model
Frontier Proprietary LLMs
Open Source Transformer Model
Frontier Proprietary LLMs
Benchmark comparison

Instruction: একটি ফাংশন লিখুন যা প্রদত্ত নেস্টেড তালিকাগুলিতে সাধারণ উপাদানগুলি খুঁজে বের করে। * তালিকা আইটেম * তালিকা আইটেম * তালিকা আইটেম * তালিকা আইটেম

Curated Instruction: একটি ফাংশন লিখুন যা প্রদত্ত নেস্টেড তালিকাগুলিতে সাধারণ উপাদানগুলি খুঁজে বের করে।

Instruction: একটি ফাংশন লিখুন যা প্রদত্ত নেস্টেড তালিকাগুলিতে সাধারণ উপাদানগুলি খুঁজে বের করে।

Translated Instruction: Write a function that finds the common elements in the given nested lists.

Given assert: ['assert common_in_nested_lists([[12, 18, 23, 25, 45], [7, 12, 18, 24, 28], [1, 5, 8, 12, 15, 16, 18]])==[18, 12]']

Curated assert: assert common_in_nested_lists([[12, 18, 23, 25, 45], [7, 12, 18, 24, 28], [1, 5, 8, 12, 15, 16, 18]])==[18, 12]

Final Instruction:
Write a function that finds the common elements in the given nested lists.
Example:

```
def common_in_nested_lists(nestedlist):
    # your code
    return nestedlist
assert common_in_nested_lists([[12, 18, 23, 25, 45], [7, 12, 18, 24, 28], [1, 5, 8, 12, 15, 16, 18]])==[18, 12]
```

Figure 1: Proposed Methodology

Model Type	Specifications	Model Name	Accuracy
Proprietary Models	Without Translation	Gpt 4o mini	43.40%
		Gpt 5 mini	55.20%
		Gpt 5	60.69%
	With Translation and Assertion	Gpt 4o mini	66.00%
		Gpt 5 mini	81.80%
		Gpt 5	83.80%
Opensource Models	Without Translation	LLama 3-8b	25.60%
		md-nishat-008/TigerLLM-1B-it	18.00%
	With Translation and Assertion	LLama 3-8b	32.40%
		md-nishat-008/TigerLLM-1B-it	21.00%

Table 2: Model Comparison

moderate, and large prompts of equivalent semantic reasoning. The resultant scores were 81.80%, 80.60%, and 80.40%, respectively, demonstrating that prompts should be concise, well-rounded, and generalizable. The System Prompts are shown in the Appendix A.

6 Error Analysis

Our analysis shows that while larger proprietary models like GPT do better than smaller open-source models, data preprocessing is still key to getting good results. Assertions improved performance by a lot, and prompt tuning increased the GPT-5 score from 80 to 83.8. Common errors were syntactic (e.g. missing brackets, incorrect indentation), semantic (e.g. translating “গণনা করো” as “print” instead of “calculate”), logical (e.g. returning a single variable instead of the full function), and hallucinations (e.g., introducing unnecessary variables). Fine-tuning reduced hallucinations and improved task generalization, but even state-of-the-art models struggled with inconsistent or ambiguous datasets. This shows that data quality and consistency are as important as model choice for Bangla code generation, and future work should focus on dataset generalization, synthetic assertions, and refined prompt tuning.

7 Conclusion

In this paper, we have highlighted the findings regarding code generation using Bangla instructions. We found that with a better and, more robust dataset, along with a concise and accurate prompt, the results of the generation can be boosted significantly. As noted, state-of-the-art and larger models have significantly outperformed open-source models. While the choice of a better model leads to better results, the gravity of a precise and factual dataset is significantly important. For future work, we will try more advanced prompting techniques and bilingual training to further boost Bangla code generation.

Limitations

Our work is limited to code generation under specific Bangla instructions. This is promising but not practical for real-world scenarios where you need deeper semantic understanding with multi-line projects. We only focus on code generation and not on everyday applications where Bangla can be translated or made accessible to non-Bangla speakers. We only benchmarked a subset of state-of-the-art models, so our comparison is not comprehensive. Hence, the results only show a part of the model's capabilities, and there is room for future work on more robust Bangla-instructed code generation.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877--1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Carlos Eduardo Andino Coello, Mohammed Nazeh Alimam, and Rand Kouatly. 2024. Effectiveness of chatgpt in coding: A comparative analysis of popular large language models. *Digital*, 4(1):114--125.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2024. mhumaneval--a multilingual benchmark to evaluate large language models for code generation. *arXiv preprint arXiv:2410.15037*.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. Tigercoder: A novel suite of llms for code generation in bangla. *arXiv preprint arXiv:2509.09101*.

Nishat Raihan, Mohammad Anas Jawad, Md Mezbaur Rahman, Noshin Ulfat, Pranav Gupta, Mehrab Mustafy Rahman, Shubhra Kanti Karmakar, and Marcos Zampieri. 2025b. Overview of BLP-2025 task 2: Code generation in bangla. In *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*. Association for Computational Linguistics.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Ahmet Üstün, Viraat Aryabumi, Zheng-Xin Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi4 Amr Kayid, Freddie Vargus, and 1 others. 2024. Aya model: An instruction finetuned open-access multilingual language model.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, and 1 others. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Abdullah Khan Zehady, Safi Al Mamun, Naymul Islam, and Santu Karmaker. 2024. Bongllama: Llama for bangla language. *arXiv preprint arXiv:2410.21200*.

A Appendix

A.1 small prompt

```
system_prompt = (
”You are a precise Python assistant.”
”Write a correct, concise solution in Python that satisfies the user's instruction. ”
”Create a generalized solution that strictly follows the given instructions and
also satisfies the assertion conditions.”
”Treat assertions as tests, not as the spec itself so that a more general solution is constructed.”
”Ensure the assertion condition is met.\n”
”<python code here>”
)
```

A.2 moderate prompt

```
system_prompt = (
”You are an expert Python assistant. ”
”Write a correct, concise Python solution that follows the user’s instruction exactly and handles common edge cases
(empty inputs, single items, negatives, large values). ”
”Treat assertions as test cases, and ensure they pass with a general solution. ”
”Avoid naive shortcuts (like suffix checks or hardcoded constants) unless explicitly required. ”
”If categorical outputs are required, return exactly the strings shown in the assertions (e.g., True/False, Yes/No,
Valid/None, Found a match!/Not matched!, Equal/Not equal, Even/Odd, Even Parity/Odd Parity). ”
”Return the correct value of the right type; otherwise return None (including on exceptions). ”
”For floats, match the decimal format in the assertion outputs. ”
”Use the exact function name and parameters given, and return the exact type/shape requested. ”
”Output only the function code (and tiny helpers if needed), nothing else.”
”<python code here>”
)
```

A.3 large prompt

```
system_prompt = (
”You are a precise Python assistant. ”
”Write a correct, concise Python solution that strictly follows the problem in the user's instruction.”
”and passes both the shown assertions and plausible hidden tests. ”
”Treat assertions as tests, not as the spec itself so that a more general solution is constructed. ”
”Ensure the assertion condition is met. ”
”Implement the natural language spec in the user's instruction. Handle standard edge cases.”
”(empty inputs, singletons, negatives, large values, mixed types) consistent with that spec. ”
”No prints, no input(), no files/network, no global state, no mutation of arguments. ”
”Ensure outputs are deterministic; if multiple valid answers exist, use a stable tie break.”
”(e.g., first occurrence or lexicographic order). ”
”Avoid oversimplified shortcuts (e.g., suffix checks, hardcoded constants, or pattern only guesses) ”
”unless explicitly required. Use logic that generalizes and covers edge cases. ”
”Be explicit about bools (exclude True/False from numeric logic unless the spec says otherwise). ”
”Avoid raising exceptions unless the spec requires it; return a neutral value instead (e.g., 0, [], None). ”
”Use exact integer math where possible; if floating point is required, define rounding/format (e.g., round(x, 2)). ”
”Use exactly the requested function name and parameters; return exactly the requested type/shape. ”
”Prefer O(n)–O(n log n); avoid unnecessary copies and heavy imports. ”
”Output only the function (and any tiny helper if essential). Do not include docstrings, comments, or extra text. ”
”Consider adversarial but spec consistent cases (empties, all equal, already sorted, duplicates, extreme values). ”
”Do not hardcode to the given assertion values. ”
”Output ONLY the function (and any tiny helper it calls). No extra text.\n”
”<python code here>”
)
```

Author Index