End-to-End Pipeline: Normalization, and Summarization of Bangla-English Code-Switching Conversation

by

Samir Rahman
24341192

Dania Siddique
24241088

Humaira Sadia Tasnim
24241055

Zahidul Islam Khan
20301158

Nayem Bin Omar
20301435

A thesis submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
January 2026

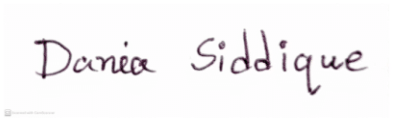
Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

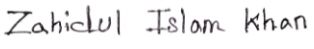
Student's Full Name & Signature:

SAMIR RAHMAN
Id: 24341192



Dania Siddique
Id: 24241088

Humaira Sadia Tasnim
Id: 24241055



Zahidul Islam Khan
Id: 20301158

Nayem Bin Omar
Id: 20301435

Approval

The thesis titled "End-to-End Pipeline: Normalization, and Summarization of Bangla-English Code-Switching Conversation"
submitted by

1. Samir Rahman (24341192)
2. Dania Siddique (24241088)
3. Humaira Sadia Tasnim (24241055)
4. Zahidul Islam Khan (20301158)
5. Nayem Bin Omar (20301435)

Of Fall, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on October 10, 2025.

Examining Committee:

Supervisor:
(Member)

Nazmul Islam
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Conventional Natural Language Processing (NLP) systems are predominantly designed and trained for monolingual text. However, the extensive use of Bangla-English and Banglish(Bengali written in Romanized alphabets) code-switching informal conversations in digital communication proves to be challenging for these NLP systems. To addresses this research gap in processing mixed language texts of Bangla-English-Banglish we proposed BiLoRA-BN, a end-to-end pipeline specifically designed for normalization and summarization of code-switched Bangla-English-Banglish conversations in digital communication. The proposed system employs a two-stage Low-Rank Adaptation (LoRA) architecture built on a shared, pre-trained Transformer as backbone with 4-bit quantization, enabling efficient multi-task learning while reducing trainable parameters. The experimental results of BiLoRA-BN are compelling, it significantly outperforms conventional sequential and cascading pipelines, with a +5.74 BLEU gain in normalization quality and a +5.26 ROUGE-1 improvement in final summary accuracy. During interface testing BiLoRA-BN also delivers results faster compared to other pipeline based cascading approach of different architecture and pre-trained models. Crucially, in the interface part, the entire system of BiLoRA-BN can operates on a consumer-grade GPUs with 8GB of memory. By directly modeling all the transformations BiLoRA-BN tries to capture the reality of multilingual digital discourse, with the complex scenario like code-switching and code-mixing in the conversations. This work contributes a step toward understanding how people naturally speak and write to communicate in digital spaces and how NLP model work with it.

Keywords: Code-Switching, Natural Language Processing (NLP), Bangla-English, Text Normalization, Abstractive Summarization, Low-Rank Adaptation (LoRA), Parameter-Efficient Fine-Tuning, mT5, Quantization, Multilingual Models, Sequence-to-Sequence Models, Computational Linguistics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the digital age, where online chat with mixed languages has become second nature, it is even more common in places with mixed-language communities, such as Bangladesh. At home or abroad, speakers of Bangla often shift fluidly in casual conversations from Bangla to English. In this daily practice, known as code-switching, individuals blend two or more languages rather than keeping them strictly monolingual. For many, that's a natural way to speak as we tend to pick the word that most closely matches the emotion or concept we want to convey, whether it's from English or any of the countless languages. However, this makes human communication easier at the same time being significantly difficult for the machines to understand them. Most natural language processing (NLP) systems weren't trained on this mixture, and they tend to struggle when confronted with tasks such as figuring out what languages are being used in a given piece of text, translating between languages, gauging the sentiment in the text or summarizing its content. This linguistic phenomenon poses a significant challenge for automated systems, as "the task of automatic language identification becomes increasingly important to facilitate further processing" in such multilingual environments [15]. These are particularly difficult when applying Bangla-English combinations, as the Bangla and English languages vary in grammar, structure. It is further compounded by the fact that Bangla words are also typed using English alphabets and this makes it even more complex by the automated technology to analyze the text.

1.2 Rationale of the Study or Motivation

In the 1950s, a new field originated which is Natural language Processing. The aim was to have machines understand and translate human language. After that, with time the evolution in this field has boosted with the advancement of artificial Intelligence and deep learning models. In the recent era, deep learning has led to a major breakthrough in NLP, powering technologies like Large Language models(LLMs). It is evident that, while these models give really good performances for monolingual texts, when it comes to other languages the resources are low. Moreover, when it comes to Bangla English code-switched texts this proves to be an extremely low resource phenomenon in NLP datasets. The general models we have now are currently not optimized for this task. "The core of the problem lies in the English-centric nature of most large-scale model pre-training. The linguistic patterns of low-resource language pairs and code-switching scenarios are severely under-represented in the training data, leading to poor performance [16]." The inability of current NLP systems to process Bangla-English code-switched data poses serious challenges to those interested in the access and understanding of the information published in the digital multilingual contexts. Particularly, text summarization is vital in deriving important points in long conversations, analysis of current trends in the public sentiment, and cross-linguistic communication. Take, for instance, an example of analyzing social media reactions about ongoing events when social media postings include both the Bangla and English language, NLP tools will be required to correctly process the hybrid linguistics contents to draw some useful insights. However, the situation gets worse by the fact that the available landscape shows a deficit in terms of specialized equipment and resources that can help struggle with this particular challenge. The same lack of technology has caused delays in the advancement of adjacent areas of NLP such as machine translation systems and conversational artificial intelligence systems. This study revolves around these weaknesses by coming up with technologies to specifically work on these linguistic code-switching. The most essential challenge is to develop systems that exhibit a greater degree of accuracy as well as being easily accessible to communities whose lingual set-up is an intermingling practice within communications. These developments would help make the language technologies more inclusive to benefit multilingual communities.

1.3 Problem Statement

Modern NLP models work poorly when used with Bangla-English code-switched conversation data. Part of this limitation is due to the fact that there is little in terms of amount and high-quality, annotated data sets that adequately reflect mixed-language communication patterns. "A primary impediment to progress in this area is the severe scarcity of high-quality, annotated datasets that capture the nuances of mixed-language communication, a well-documented bottleneck in code-switching research [17]." There are also innate difficulties of bilinear processing of two linguistically varying systems simultaneously. It is specifically observable with the tools of summarization in the dialogue with code-mix where there is significant weakness in terms of retention of the semantic meaning. The existing workaround methods often result in summaries with no or inaccurate contextual information or the wrong interpretation of the purpose of the original material. "The challenge is acutely evident in dialogue summarization, where models struggle to retain the semantic meaning and contextual integrity of the original code-mixed conversation, often producing incoherent or factually inaccurate outputs [18]." The present thesis resolves such issues by proposing an overall end-to-end pipeline that can engage in both normalization and summarization processes of code-switched and mixed Bangla-English conversations.

1.4 Objectives

The proposed research will take the form of the following key objectives:

- The research will first undergo a detailed analysis of the existing practices of normalisation of the text of code-mixed Banglali and English, botanically recording their specifics of strengths and weaknesses. The analysis discussed will be a foundation of perceiving what the modern condition of the field is and what should be changed.

- This shall then update and make alterations on existing summaries models, that their liquidity towards mixed-language content processing may be improved. This will bedone through the analysis of various architectural strategies and locating the modifications.that will bring the optimum performance to the code-switching text.

- The key input is the development of an integrated processing pipeline that combines normalization and summarization with custom functionality used upon. Bangla-English conversations. The latter will make sure that this system does not. undermine the performance of mixed-language content.

- The study will also take stringent evaluation plans in order to evaluate the output. of the proposed pipeline, a comparison to the existing tools regarding the appropriateness measures. This will ensure how effective the system is on various factors. classifications of code-switched contents.

1.5 Methodology in Brief

The systematic, pipeline driven approach will be employed in the methodology of conducting the research to build up an end-to-end system normalization and summarization of code-switched conversations in Bengali and English. The process consists of six consecutive stages General Problems Exploration, Data.Preparation, Model Selection & Training, Model Evaluation, System Integration and Deployment & Scalability.

The first phase is conducting literature review. In this phase research motivation, objectives and problem statements are defined. Afterwards, analyzing existing solutions and research gaps are identified. The second phase involves the collection and refinement of a code-switched dataset. Exploratory Data Analysis (EDA) will be done to understand mixing patterns and challenges. Additionally, Data Preprocessing and Normalization step will be done for cleaning, transliterating Romanized Bangla (Banglish) to the native script, and correcting spelling variations. After that, the preprocessed dataset will be split into training, validation, and test sets. Phase 3 is Model selection & Training. We will select pre-trained models (e.g., mBERT, mT5, XLM-R) for both the normalization and summarization tasks. Moreover, the model training involves fine-tuning on our preprocessed datasets. Later on, we can monitor training metrics like loss and accuracy to prevent overfitting. In Phase 4 which is model evaluation (BLEU, WER, CER, Qualitative Analysis).Then plot training and error analysis will be done and we will compare it to baseline methods. Phase 5 is System Integration. The individually developed normalization and summarization modules will be integrated into a single, coherent pipeline. In the last phase which is Deployment & Scalability we will evaluate the complete pipeline.

The system's performance will be compared to the baseline methods using both automated metrics (e.g., BLEU, ROUGE) and qualitative error analysis to identify failure modes. Lastly, the system will be prepared for deployment. The research will conclude with the publication of results and recommendations for future work. The final pipeline is implemented as BiLoRA-BN, a shared-backbone system that integrates normalization and summarization in an end-to-end workflow.

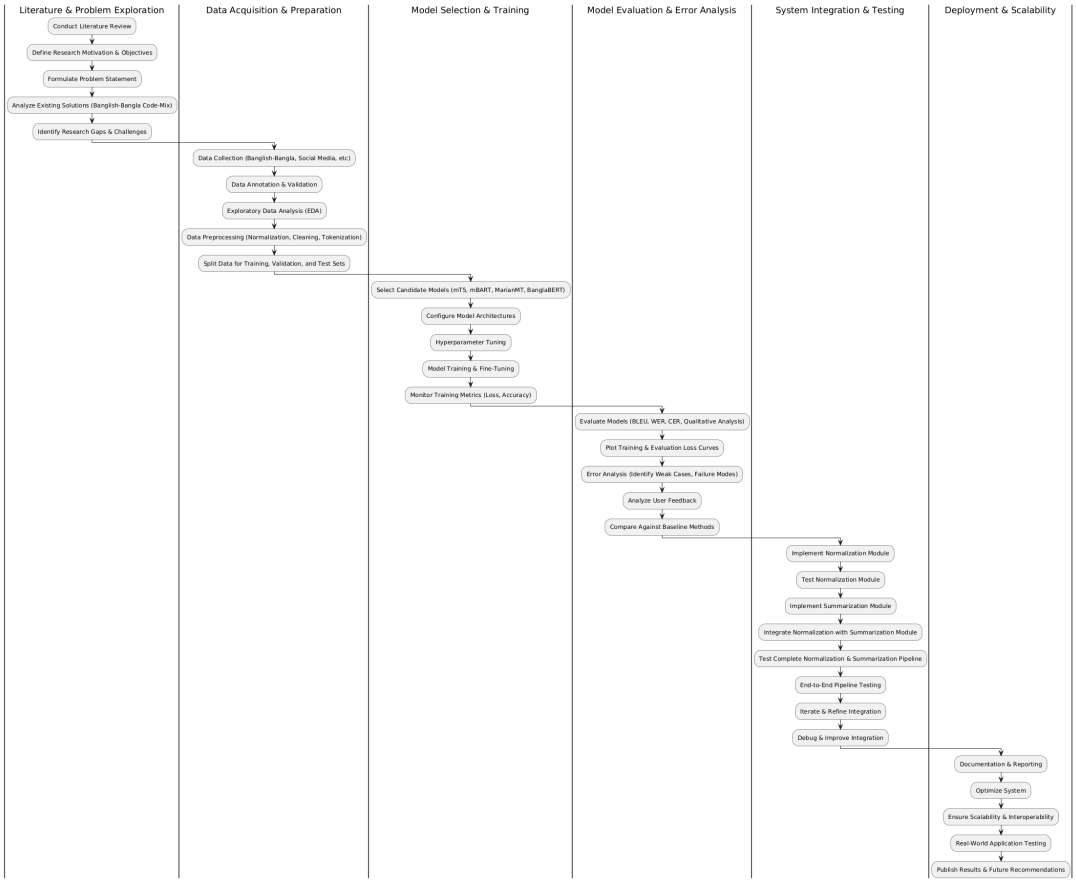


Figure 1.1: System architecture overview

1.6 Scope and Challenges

Scope

This study is specialized in informal code-switched Bangla-English text that is received in the free social media sources, such as Facebook and Twitter. On the one hand, the research will focus on the texts that show the equal use of the two languages and avoid more or less monolingual text with only slight code-switching phenomena. The technical approach is to use sequence to sequence models with attention mechanisms both in

normalization and the task of summarization. It is an architectural decision which uses current best practice in neural language processing, but allows the flexibility to deal with mixed-language content as well.

Challenges

There are critical issues and challenges are expected to come up in the course of this experiment, few to mention are as follow:

- **Data Scarcity:** Annotated Bangla-English code switched materials is very difficult to find specifically in the context of dialogues and informal conversations.
- **Linguistic Complexity:** To satisfy the variation of code-mixing variants, such as uneven transliteration, change of structure of words and vague syntax.
- **Model Generalization:** This feature is a measure of the ability of a system to perform outside of training data on the non training conversations over in the real world.
- **Evaluation Metrics:** It needs to put in place a sufficient means of defining the quality metrics which are practically effective in a manner such that they serve in the activity of quantifying the degree to which the system normalizes and summarizes the code-mixed text.

Chapter 2

Literature Review

2.1 Preliminaries

Code switching is widely common nowadays. People who know multiple languages often move between languages when they speak and write. This language is not used randomly, it usually follows certain rules that are dependent on social, contextual, and grammatical considerations. And this makes it an essential field of research in natural language processing (NLP) and computational linguistics. We can define Code-switching as the use of more than one language within a single conversation, sentence, or phrase. It has a number of social and communication purposes, it helps us to easily communicate and adapt to vocabulary limits. Code-switching (CS), the alternation between two or more languages within a single conversation or utterance, is a common and rule-governed practice among multilingual speakers [17].

At first sight, the code-switching practice may seem informal or unseemly yet the code-switching is a cautious and propositional method. Although not a random phenomenon, it is hampered by grammatical, social, and contextual aspects and thus a crucial field of research in computational linguistics and Natural Language Processing (NLP) [18]. The reasons for code-switching are ease of expression, emphasis and identity signaling. The individuals may even change languages to convey an idea, thought, and emphasize a point in a better way. In other scenarios, code-switching is used to create solidarity, unity or to create group membership such as they are able to speak the same languages [19]. It may also indicate a shift of the topic focus or degree of formality, especially in multilingual societies whereby different languages have been linked to different spheres

of application.

As well, code switching has various varieties depending on linguists and is determined by the point where the language change takes place. And one of them is Inter-sentential switching of codes on the sentence boundaries. As one example, a speaker can complete a phrase in Bangla and then proceed to English. This style is more systematic and is usually applied by the fluent speakers of the two languages. Conversely, intra-sentential code-switching occurs within the same phrase or clause, which points to a freer grammatical mixing. This type is more complex and characteristic of the people who feel quite at ease using both languages. And there is an alternate variant that is tag-switching, in which a tag phrase of one language is put into a sentence otherwise in another, such as with you know or I mean in a Bengali sentence. These tags have discourse purposes and are simple to insert without causing problems to the grammatical sentence composition.

The concept of normalization in NLP is converting informal and noisy text into a uniform and standard form. This is particularly acute when it comes to texts coded-switched or informal, which can include inconsistencies in the spelling and script, as well as the use of the language[4]. The downstream NLP tasks of the normalization can be achieved through effective normalization translation, parsing, summarization.

Text summarization is a task required in NLP and is a task that requires the generation of a short version of a longer text with no alteration of its real meaning. There are two approaches predominant to summarization namely extractive and abstractive. Text summarization is used to produce a summarized form of the source text in a distilled and fluent form, and reflects the essential meaning of the original textual content. There are wide-ranging approaches, which can be divided into two categories [21].

To conclude, it is clear that the basic concepts of code-switching, normalization, and summarization form the basis of successful NLP systems that can be created to consider multilingual settings. The form in which this is done by the Normalization process makes the data to be in an accessible form and it has detected the problems such as transliteration, translation, language recognition and spelling mistakes. Extractive or abstractive: Summarization provides a way to summarize information, but at the cost of some difficulties in the multilingual and informal texts. These underlying issues are essential in the context of building sound and successful NLP code-switching applications in Bengali English code-switched conversations.

2.2 Review of Existing Research

Forming a complete pipeline on normalization and summarization of a written Bangali conversation that involves the use of English would entail a critical review of numerous areas of interdependent study. According to the leading literature, the results of the review will combine information on the linguistic history of code-switching (CSW), evolution of computational solutions to work it, technical peculiarities of the lexical normalization, abstractive dialogue issues summarization, and the resources of Bangali-English language pair.

Clothing the core of it, the research paper which was done by a linguist named Shana Poplack in 1980 speaks about it. Code switching in regard to where in the sentence structure that the code switching actually occurs. Poplack's framework is based on analysis of 1,835 code-switched utterances from 20 Puerto Rican speakers, positioning three main types of code-switching, governed by a stringent "equivalence constraint" and a "free morpheme constraint." [17]

The field of computational code-switching is not new. It has a rich history and throughout many evolutions today's NLP models do a decent job in this scenario. This journey, systematically mapped in the survey by Winata et al. (2022), is a direct procession from linguistic-based approaches to the giant AI models of today [2]. The invention of the Transformer architecture and Pre-trained Language Models (PLMs) like BERT promised a new era. However, a critical and systematic study by Winata et al. (2021) provided a crucial reality check, showing that multilingual models have limitations for code-switching specific tasks [3].

In constructing the available framework of lexical normalization, the paper by van der Goot & Ã‡etinoÄŸlu (2021) is exceptionally comprehensive, relevant as the initial step in the processing of informal non-standard text of the Code-mixed variety [4]. The evidence presented in the paper supports the fact that a specific normalization step directly benefits downstream tasks with relative performance gain of 5.4 percent on Part-of-Speech (POS) tagging.

The SAMSum Corpus, introduced by Gliwa et al. (2019), was a landmark because it provided over 16,000 human-written summaries of messenger-style chats, giving researchers a perfect benchmark [8]. The particular issue of code-switched conversation summarization has been covered by Mehnaz et al. (2021), where authors present the

dataset GupShup over Hindi-English [9].

2.3 Summary of Key Findings

The extensive literature review shows that there are a number of key findings, which altogether determine the state of the art and specify the research gap the present thesis addresses.

- **Code-Switching is Syntax-Based in Essence:** Research has clearly defined code-switching as a behavior governed by syntax. Modern computational experiments have demonstrated that syntactic structure is sufficient and a very strong predictor of natural CSW [1].
- **General Models Are Not Always Superior to Specialized Ones:** The "bigger is better" idea of general NLP cannot be applied to CSW completely. Special-purpose architectures that specifically attempt to approximate language-mixing process or syntax structure are likely to offer better performance and efficiency trade-offs [3].
- **A Hybrid Pipeline Architecture is Needed to Normalize CSW:** Normalization of informal, code-switched text is a many-stage, difficult task which demands language recognition, phonetic normalization, and morphological parsing [4], [5].
- **Dialogue Summarization is a CSW Double Jeopardy task:** Summarizing conversations itself is not easy, with information being scattered across multiple turns, with the additon of code-switching, goal of this experiment become more daunting. [9]. Moreover, common evaluation methods such as ROUGE are not much reliable in this task [8].
- **Identifying the Research Gap:** No existing work has designed an end-to-end pipeline to accept raw, informal code-switched conversational text, normalize it properly, and extract an abstractive summary. The Bangla-English language pair is still computationally under-ready [6].

Chapter 3

Requirements, Impacts and Constraints

This chapter talks about the hardware, software, and data requirements needed to build and run our BiLoRA-BN system. We also discuss the impacts our work can have and the constraints we faced during this project.

3.1 Final Specifications and Requirements

To develop the BiLoRA-BN pipeline (Banglish→Bangla normalization followed by Bangla summarization), we needed specific hardware, software tools, and datasets. Below we list out what was needed for both the training phase and the inference (testing) phase.

3.1.1 Hardware Requirements

Training a large Transformer model with LoRA adapters requires good GPU resources, but thanks to 4-bit quantization, we managed to keep it accessible on consumer hardware. For inference, the requirements are much lower.

For Training:

- **CPU**: For the training purpose, a modern multi-core CPU, like an Intel i7/i9 or AMD Ryzen 7/9 equivalent is needed, to handle data preprocessing and tokenization.

- **GPU**: GPU is the key component in training phase. The BiLoRA-BN system's weights were fine-tuned using RTX 4090 with 24GB of VRAM.
- **RAM**: To manage batch preparation and dataset loading, especially when managing longer sequences a minimum of 32GB of system RAM is needed
- **Storage**: Here SSDs is preferable as disk storage type for faster data loading and training of weights. Around 50GB disk space is needed for model checkpoints, logs and datasets during training.

For Inference:

- **CPU**: When it comes to inference, the needs are much lower. The process is mostly GPU-bound, so any modern CPU will suffice.
- **GPU**: A GPU with 8GB of VRAM is enough to run the quantized model smoothly.
- **RAM**: In interface stage, 8GB system RAM is enough for to manage the computation.
- **Storage**: Roughly 5GB of SSD disk storage is needed to save the quantized base model and adapter weights.

It's worth highlighting that without 4-bit quantization, the full model fine-tuning would require over 40GB of VRAM for training. And the training process would take days. Using QLoRA is what made this experiment feasible.

3.1.2 Software Requirements

The entire pipeline and all the experiments were done using Python and open-source machine learning libraries. Below the software stack:

Operating System:

- Linux (Ubuntu) for compatibility with CUDA and PyTorch.

Core ML Libraries:

- **Python 3.12+**: The core programming language for all scripts.

- **PyTorch 2.0+**: Deep learning framework for model training and inference.
- **Hugging Face Transformers 4.36+**: For loading pre-trained mT5 models and tokenizers.
- **PEFT (Parameter-Efficient Fine-Tuning) 0.6+**: For LoRA adapter implementation.
- **BitsAndBytes 0.41+**: For 4-bit quantization (NF4 format).

Evaluation and Utilities:

- **ROUGE (rouge-score)**: For summarization evaluation metrics.
- **SacreBLEU**: For BLEU score calculation in normalization evaluation.
- **JiWER**: For Word Error Rate (WER) and Character Error Rate (CER).
- **Pandas, NumPy**: For data manipulation and analysis.

Documentation and Writing:

- **LaTeX (XeLaTeX)**: For thesis writing with Bangla font support.
- **Noto Serif Bengali**: Font for rendering Bangla text in the thesis.

3.1.3 Data Requirements

The experiments requires two parallel datasets: one for normalization (Banglish→Bangla) and one for summarization (Bangla conversation→Bangla summary).

Normalization Dataset (Adapter-1):

- Banglish to Bangla, annotated pairs at word, sentence, and paragraph levels.
- Total: 34,443 pairs (27,075 words + 7,258 sentences + 110 paragraphs).
- Source: Derived from publicly available Romanized Bangla Corpus.
- Split: 90% training, 5% validation, 5% test.
- Preprocessing: In this process, pure English words were removed, different spelling variations were standardized, and low-quality pairs were filtered out.

Summarization Dataset (Adapter-2):

- Bangla conversation to summary pairs for dialogue summarization purposes.
- Total: 16,369 pairs derived from SAMSum corpus which were then translated to Bangla.
- Additional: 250 manually curated Banglish→Bangla summary pairs were made for end-to-end evaluation.
- Domain: Mostly informal conversations (daily life, planning, discussions).
- Split: 90% training, 5% validation, 5% test.

Data Quality Considerations:

- Inconsistent Banglish spelling is a major challenge which had to be standardized. (e.g., "ami", "aami", "amii" all mean "I").
- English words mixed in Banglish were a big challenge, since real conversations have code-mixing.
- Manual review was done on a subset to ensure the translation quality.

3.2 Societal Impact

The BiLoRA-BN system can have huge impacts on society, especially for millions Bangla-speaking communities who communicate on a mixed code-switching messages:

- **Benefits for Bangla Speakers**: Over 230 million people speak in Bangla, the BiLoRA-BN system will help to bridge the gap by processing code-mixed text that people actually use in day-to-day online communication.
- **Accessibility for Non-Native Typists**: Many non-native bangla speaker find typing in the Bangla script difficult instead they type in romanized alphabets. BiLoRA-BN, normalization component makes it possible to convert such text to proper Bangla, which will make the digital communication more accessible for everyone.

- **Educational Applications**: As many of Gen-Z generation prefer typing in a mixed language, the BiLoRA-BN system can be used in educational settings will help them to learn proper Bangla writing by seeing how their Banglish text converts to standard Bangla.

- **Customer Support**: A very real use case of BiLoRA-BN system is in online businesses specially where they can use the summarization feature to quickly understand customer conversations written in Banglish, improving response times and service quality.

- **Risk of Misinformation**: With all these positive impacts there is one thing to be concern about, which is, if the summarization produces by BiLoRA-BN incorrect then it can spread misinformation that can mislead.

All in all, by bridging the gap between informal communication and formal communication, the BiLoRA-BN system can empower millions of users by supporting learnings, and helping businesses to connect more effectively and efficiently.

3.3 Environmental Impact

Training large language models need huge computation which lead to high energy consumption. To reduce the energy consumption and environmental footprint, several steps were taken:

- **LoRA vs Full Fine-Tuning**: Instead of fine-tuning the entire massive model from scratch, an efficient method called LoRA has been used. With the use of LoRA method only a tiny fraction (a mere 0.5%) of the model's parameters have been trained, cutting the computational effort for training by about 99.5%.

- **4-bit Quantization**: To make the model run leaner and faster with less computation, a technique called 4-bit quantization have been used. This shrank and compress the model's memory needs dramatically, leading less calculations.

- **Estimated Carbon Footprint**: In practical terms, final training run for both adapters took about 10 hours on a single RTX 4090 GPU. We estimate it used

roughly 2.25 kilowatt-hours of energy, which is way less than training all 3.7B parameters.

- **Checkpoint Reuse**: Throughout all the experiments, multiple checkpoint has been used to save intermediate developments during training.
- **Minimizing Training Runs**: Then number of epochs was planned carefully to avoid redundant runs, training each adapter just a couple of times to get it right.

3.4 Ethical Issues

All the experiments of this study was guided by a strong commitment to ethical and responsible development. Some ethical considerations that has been considered during this development:

- **Data Privacy**: Data privacy was the first and foremost priority as the experiments done in this study works with personal conversations for training purpose. We ensured that the SAMSum and Romanized Bangla Corpus datasets has followed all the established standards.
- **Consent**: For the SAMSum-derived dialogue summarization data, the original dataset was collected with proper consent from participants by following all the protocols.
- **Anonymization**: Any identifiable information such as locations, name and phone numbers has been rigorously anonymized by placeholders or fictional equivalents.
- **Bias Risks**: Transformer models learn from the data they're given. For example, if the training data contains more male oriented dialogue than female, the model might perform better on male-associated text.
- **Hallucination Risks**: Like all other NLP text generation systems, this summarizer system can sometimes produce information not present in the input (hallucination).
- **Safe Usage Boundaries**: The BiLoRA-BN, is specially trained on informal conversational text. Thus, it should not be used for legal, medical, or other high-stakes document processing without careful validation.

3.5 Project Management Plan

The whole research and development process was structured in three phases, as below:

Phase 1: Data collection and problem formulation

The work in this phase primarily covered problem analysis, definition, identifying research gaps in the field, and assembling high-quality datasets for further system modeling.

Below mentioned activities were carried out at this phase:

- Related literature review on Banglish–Bangla normalization as well as dialogue summarization
- Problem definition and requirement analysis
- Banglish–Bangla parallel corpus collection and curation
- Initial data cleaning and exploratory analysis

This phase aimed to confidently establish the data and task scope prior to developing the system.

Phase 2: Data preprocessing and creation of Banglish-to-Bangla normalization module

The second phase focused on processing raw data into model-specific training and evaluation datasets, which led to the development of the Banglish-to-Bangla normalization system component, a pipeline step feeding into the final end-to-end system.

Some of the activities proposed for the second phase were:

- Data preprocessing and normalization (Unicode standardization, noise reduction, consistency check among tokenization)
- Selection of the suitable transformer architecture
- Baseline normalization models were created
- Adapter-1 training for Banglish-to-Bangla normalization was carried out
- An initial evaluation of the normalization performance was achieved

The second phase led to the creation of a strong normalization module that appears as an initial pipeline step, responsible for normalization.

Phase 3: Bangla dialogue summarization pipeline integration

The concluding phase concentrated on building the end-to-end system pipeline, linking the normalization in the preceding step to Bangla dialogue summarization.

Some of the activities earmarked for this phase were:

- Baseline end-to-end pipeline modeling
- Adapter-2 training for Bangla dialogue summarization
- Integration of normalization and summarization into a single framework (BiLoRA-BN)
- Comprehensive evaluation of the component and end-to-end performance
- Final report writing and analysis of results

This step resulted in the packaging of the finalized BiLoRA-BN system that could directly turn Banglish dialogue into intelligible Bangla summaries.

Figure 3.1 illustrates the high-level workflow of the proposed project plan.

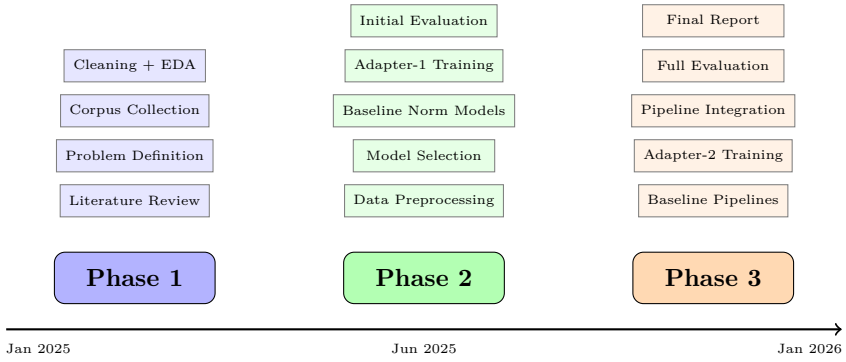


Figure 3.1: Project Plan: High-Level Three-Phase Workflow

3.6 Risk Management

All research projects come with risks that need to be evaluated and addressed accordingly. For this project, prior to executing all planned tasks, we anticipated a series of potential roadblocks that may arise during the development of BiLoRA-BN and proposed solutions for each. One of the main issues was the possibility of running out of GPU memory while

we were training the model. In order to resolve this, we chose to implement 4-bit NF4 quantization with QLoRA, which helped VRAM consumption remain under the threshold of the 16 GB capacity GPU of the RTX 4090. With regard to risk in the dataset, we predicted the inconsistent spelling of Banglish text and the small size of available code-switched corpora. However, we were able to reduce this risk by conducting careful prepossing, manual inspection of a representative subset, and constructing datasets of varied granularity (word, sentence, and paragraph levels) to maximize the diversity of the training signal.

3.7 Economic Analysis

This section provides a brief cost analysis of the whole experiment and probable cost of reproducing the results of this study:

Hardware Costs:

- **GPU Training**: The experiments used RTX 4090 GPU during whole training process. Without local access, comparable cloud instances on platforms like AWS or Google Cloud typically cost between $1 to $2 per hour.
- **Storage**: The required storage for model checkpoints and datasets came to about ~50GB. In this case, cloud storage costs are negligible (<$1/month).

Software Costs:

- On the software side, every tool used in the process, is free and **open-source**. This includes our core programming environment (Python, PyTorch), the Hugging Face Transformer models, PEFT libraries for model development, and all evaluation metrics.
- The documents was prepared using LaTeX (TeX Live), which is also freely available.

Cost Savings from Our Approach:

- **LoRA vs Full Fine-Tuning**: Full fine-tuning of the BiLoRA-BN'S 3.7B parameter would require 40GB+ VRAM (A100 GPU, $3-4/hour) and 50+ hours of training. Implemented QLoRA approach saved an estimated $200-300 in compute costs if compared with cloud computing cost.
- **4-bit Quantization**: This quantization reduces the memory requirements by 4x, enabling use of consumer hardware instead. This approach also reduce computation and provide faster interface speed.
- **Adapter Weight Storage**: Each adapter is around **100MB** instead of multiple GB, reducing storage and transfer costs.

Total estimated cost of reproducing the results of this experiment is under $50 for cloud resources (if local GPU was not available), with all software being free. This demonstrates that our approach is economically feasible for small research teams.

Chapter 4

Proposed Methodology

4.1 Design Process or Methodology Overview

This chapter describes the end-to-end methodology used to build the BiLoRA-BN pipeline, from data collection and preprocessing to model training, evaluation, and inference. The workflow integrates Banglish→Bangla normalization with Bangla summarization in a single shared backbone using LoRA adapters and task-specific prefixes.

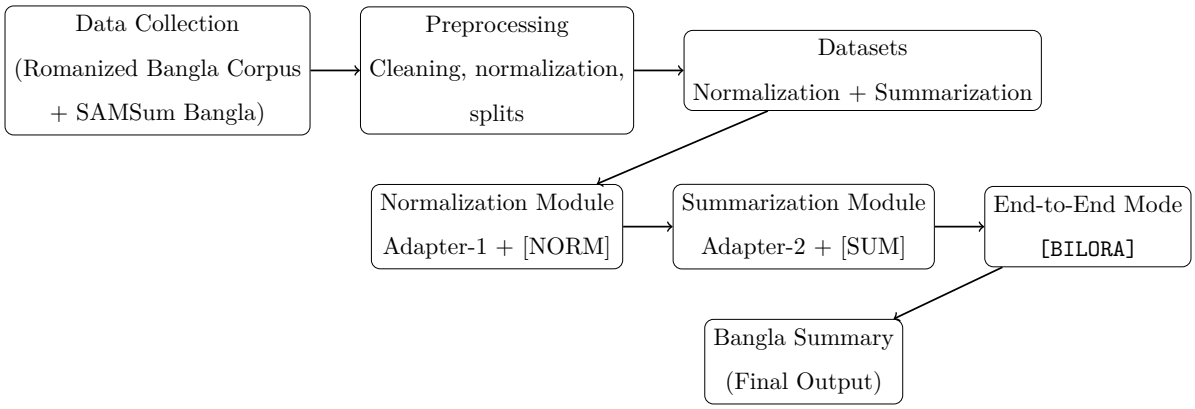


Figure 4.1: End-to-end BiLoRA-BN pipeline from data collection to final summary generation

4.2 Preliminary Design or Design (Model) Specification

Multiple baseline models were implemented for normalization and summarization: mBART, MarianMT, mT5-base, and BanglaBERT. These are sequential encoder-decoder Transformer models trained on the cleaned datasets for Banglish→Bangla normalization before moving to end-to-end experimentation.

4.2.1 Model Categories

Understanding the three main categories of the models we examined and the significance of each category for our Banglish to Bangla task is helpful before diving into the results. These categories are illustrated graphically in Figure 4.2.

Monolingual Models (Bangla-specific): These include **BanglaBERT** and **BanglaT5**. They were pre-trained almost entirely on Bangla text, so they know the language well. The idea was that maybe a model that really understands Bangla grammar and vocabulary could do a better job at producing clean Bangla output. However, a clear gap still exists because mBART's pre-training did not include Romanized or code-switched Bangla. In other words, it learns English and Bangla separately, but the mixed "in-between" from that Banglish represents remains less familiar. mT5-base and MarianMT were trained on 100+ languages at the same time. The high exposure provides them to see different kinds of scripts, language and styles. For example, mT5 has a shared multilingual vocabulary through a SentencePiece tokenizer that breaks down words into sub-word pieces to cover unseen words, which is very necessary in Banglish, as the spelling is always inconsistent and noisy. Having ability to break down the words and correctly represent the non-predictable words is very important.

Multilingual Models: mBART, mT5-base, and **MarianMT** fall into this category. **mBart** is a multilingual model pre-trained on 25 languages, including English and Bangla. We expected that both scripts during pre-trainning might help it correct the Latin letters used in Banglish with the Bangla script we want as output. We fine-tuned mBART on our normalization and summarization dataset. To some extent, it does outperform monolingual baselines, likely because it is already comfortable with both scripts. However, a clear gap still exists because **mBART's** pre-training did not include Roman-

ized or code-switched Bangla. In other words, it learns English and Bangla separately, but the mixed "in-between" from that Banglish represents remains less familiar. **mT5-base** and **MarianMT** were trained on 100+ languages at the same time. This wide exposure means they have encountered many scripts, writing patterns, and language structures. mT5 in particular benefits from a shared vocabulary across languages, using a SentencePiece tokenizer that can split unfamiliar words into smaller subword units. This flexibility becomes important for Banglish, where spelling is often inconsistent and noisy, because the model can still break down and represent messy tokens rather than failing on them as unknown words.

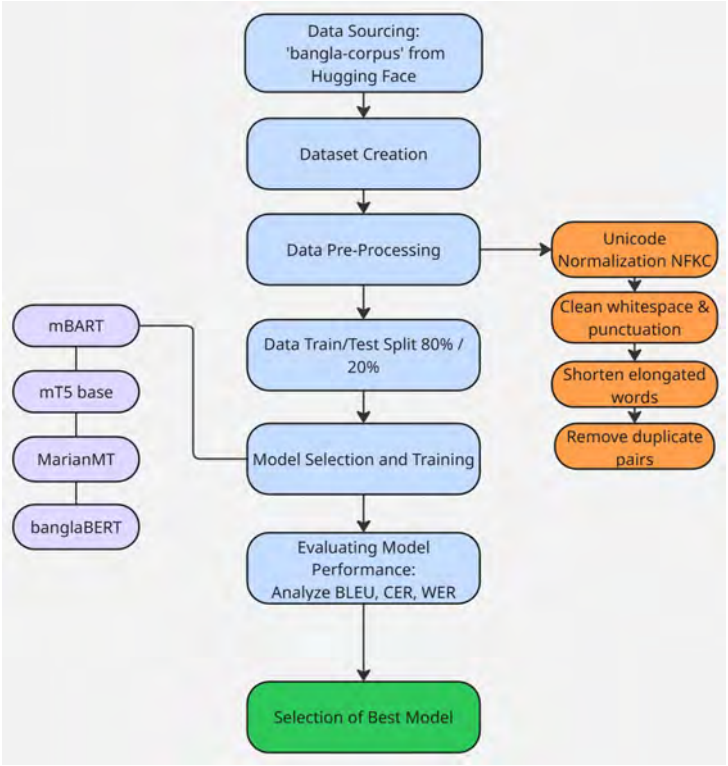


Figure 4.2: provides an overview of the evaluated methodologies, grouping the baseline models into two categories: Monolingual (Bangla-specific) models and Multilingual architectures.

4.2.2 Training Setup for Baselines

To ensure a fair comparison, we used an identical training setup for all models. Each one was fine-tuned with identical data splits: 90% for training, 5% for validation, and 5% for testing. The training set combined our word-level pair (27,075 sample), sentence-level pairs (7,258 samples), and paragraph-level pairs (110 samples). All samples were shuffled together so that the models consistently saw a mixture of short and long sequences during

training. **Architecture Details:**

All the models we tested follow the encoder-decoder architecture, which is standard for sequence-to-sequence tasks. Encoder takes the Banglish input and compress it into a sequence of hidden vectors. These vectors encodes the semantics of the input in a form that the model can process. The decoder takes these vectors as input and produces the Bangla translation one token at a time.

Figure 4.3 shows this general flow. The input goes through tokenization first (breaking text into subword pieces), then through the encoder layers, then through the decoder layers, and finally through a softmax to predict each output token. The attention mechanism lets the decoder look back at different parts of the input when generating each output token – this is important for our task because the model needs to know which Banglish word it's currently translating.

Aspect	mBART	MarianMT	mT5 (base)	BanglaBERT (Custom designed Encoder + Decoder)
Model type	Multilingual denoising encoder+decoder (seq2seq)	NMT model (Marian) often trained per language pair	Text-to-text encoder+decoder (T5-style) for many tasks	Custom made Encoder+Decoder model specialized for Bengali
Pretraining objective	Multilingual denoising	Supervised NMT training on parallel corpora	Text-to-Text pretraining	Token-detection focused on Bangla
Primary use-cases	Translation, multilingual generation, seq2seq fine-tuning	Practical pairwise translation for specific pairs	Translation, summarization, QA, and flexible in text-to-text tasks	Mainly for classification, NER, QA. Not optimized for free-form generation
Language coverage	Multilingual (mBART-50 covers ~50 languages)	Bilingual mostly (many language pairs across the family)	Very broadly Multilingual (~100+ languages in mT5 variants)	Bangla only but tried to make it a bilingual NMT model
Strengths	Good seq2seq pretraining for low-resource fine-tuning	Small/efficient pair models	Unified text-to-text interface with broad multilingual transfer	Strong Bengali understanding due to focused pretraining
Limitations	Heavier than single-pair models	Not a single universal model and quality depends on pair data	Large; may mix languages if not constrained in decoding	Not suitable for general multilingual generation tasks

Figure 4.3: Model Overview: Encoder-decoder architecture used across all baseline models. The encoder processes Banglish input into hidden representations, and the decoder generates Bangla output tokens

Training Hyperparameters:

We tried to maintain hyperparameters consistent wherever possible. Every model

used were trained for a maximum of **10 epoch** with **early termination** determined by validation loss (patience of three epoch). Every model had the same **learning rate** at **2e-5** with a **linear warm-up** in the first 10% of training steps. Due to GPU memory constraints, we used a batch size of 8 for a larger model like mT5 and mBART and 16 for smaller models like BanglaBERT and MarianMT. The AdamW optimizer and a weight decay of **0.01** were used for training. The fact that tokenization varied between models is a crucial aspect. The same Banglish word can be segmented differently depending on which tokenizer is utilized because each model employs a separate tokenizer that was taught during pre-training.

4.2.3 Qualitative Comparison of Model Outputs

We thoroughly examined each model's output before getting into the score. This qualitative analysis looks at aspects that the matrices don't completely reveal, like the kinds of mistakes the model frequently makes and whether those mistakes are minor spelling mistakes or more major miscommunications. Figure 4.4 presents side-by-side outputs from multiple models on the same test inputs. From these examples, a few clear patterns emerged:

Monolingual Model Behavior:

BanglaBERT & **BanglaT5** performed the worst on our task. Since they were trained on Bangla script, the model was confused once seeing the romanized input. At times, they gave outputs that appeared to be Bangla but were rendered nonsensical by the model even though it had produced it entirely from valid tokens in Bangla.

One specific thing we saw from monolingual models was that sometimes, these models would "hallucinate" Hanglish words/phrases that were unrelated to the input. They would just start producing common Bengali phrases that themselves were irrelevant to the input. The model instead produced the output they were most likely to produce during the pretrain phase. This behavior was more prevalent in longer inputs where the model lost track of what it was trying to produce.

mBART Behavior:

mBART performed better than the monolingual models but still had its quirks. The good news was that it rarely produced completely nonsensical output – it seemed to

understand that the task was to convert one form of text to another. The bad news was that it sometimes confused similar-sounding words, producing Bangla words that sound similar to the intended word but have different meanings.

For example, in some instances of the input containing ”bari” (home), mBART simply translating to bangla words with the same sound patterns such as ”bari” for ”outside” or ”jol” (water). This may have been an indication of the model attempting to use sound cues rather than learning their actual meaning. It also sometimes failed to translate an English word in the middle of a string of Banglish, more specifically more common English words such as ”office” or ”phone”.

Multilingual Model Behavior:

mT5-base and **MarianMT** showed the most consistent performance. **mT5** in particular seemed to handle the task well – its outputs were usually grammatically correct Bangla and preserved the meaning of the input. When **mT5** made mistakes, they tended to be minor: a slightly different word choice that still conveyed the same meaning, or occasional spelling variations in the Bangla output.

Since MarianMT was trained for translation, it knew the input-output mapping quite well. Yet we found its translations to be a bit too literal at times and a little stiff - they didn't feel quite as natural in Bangla. The outputs of mT5 felt more natural in comparison—we think its txt-to-txt pretraining might have given it more experience with different kinds of generation tasks.

Level	Banglish Input	BanglaBERT Output (Worst)	MarianMT Output	mBART Output	mT5-base Output (Best)	Analysis
Word	chatro	চাত্র	শিক্ষার্থী	ছাত্র	ছাত্র	BanglaBERT misspells. MarianMT gives a related but different word. mBART and mT5 are perfect.
Word	school	সকুল	বিশ্ববিদ্যালয়	স্কুল	স্কুল	BanglaBERT misspells. MarianMT hallucinates a related word. mBART and mT5 are perfect.
Word	accessibility	এক্সেসিবিলিটি	সহজলভ্যতা	অ্যাক্সেসিবিলিটি	প্রবেশগম্যতা	BanglaBERT & mBART only transliterate. MarianMT gives a good translation. mT5 provides the most accurate and contextually correct translation.
Short	amar thesis project shesh korte hobe.	আমার থেসিস প্রজেক্ট... (incomplete)	আমার থিসিস প্রজেক্ট শেষ করতে হবে।	আমার গবেষণা প্রকল্প শেষ করতে হবে।	আমার থিসিস প্রকল্পটি শেষ করতে হবে।	BanglaBERT fails. MarianMT & mBART translate well, but mT5 produces the most natural and fluent sentence structure.
Medium	ei model ti banglish theke bangla onubad...	এই মডেল টি... (fails)	এই মডেল টি বাংলা থেকে বাংলা উচ্চারিত করার জন্য...	এই মডেল টি বাংলা থেকে বাংলা অনুবাদ করার জন্য...	এই মডেলটি বাংলিশ থেকে বাংলায় অনুবাদের জন্য...	BanglaBERT fails. MarianMT makes a semantic error. mBART is correct. mT5 is perfect and uses the most polished grammar.
Large	Kuet moitri hole vote shurute deri...	Output Fails	Output incomplete / Fails to process	কুয়েত মৈত্রী হলে ভোট শুরুতে... (Loses coherence)	কুয়েত মৈত্রী হলে ভোট শুরুতে দেরি সকল জল্পনা-কল্পনার... (Degrades after ~20 words)	Lower-tier models fail completely. mBART loses coherence quickly. mT5 translates a significant portion correctly before degrading, showing better but still imperfect long-context handling.
Large	Sgt. Zahurul Hoque holer samne...	Output Fails	Output is truncated	সার্জেন্ট জহুরুল হক হলের... (Partial and unnatural)	সার্জেন্ট জহুরুল হক হলের সামনে ভোটার শিক্ষার্থীদের লম্বা লাইন দেখা গেছে... (Truncated)	mT5 processes more of the input accurately than mBART before hitting its context limit, demonstrating superior but not flawless long-form translation.

Figure 4.4: Model Performance Comparison: Side-by-side comparison of outputs from different models on the same Banglish inputs. Monolingual models struggled with Romanized input, while multilingual models (especially mT5) produced more accurate and fluent Bangla

Error Patterns Summary:

We clustered the main errors observed in our qualitative analysis into the following groups:

- **Script confusion:** Model does not read Romanized input correctly (predominantly monolingual).
- **Phonetic errors:** Sounds like the original input, but is incorrect in meaning (predominantly mBART).
- **Untranslated tokens:** English words present in output that were not in input (present in all, but fewest for mT5).
- **Truncation:** Output is too short, information is lost (predominantly monolingual).

- **Hallucination:** Additional information is added to the output which was not present in the input (predominantly monolingual).
- **Literal translation:** Syntactically correct, but linguistically unnatural. (predominantly MarianMT)

Using the above, we gained an insight into not just the performance of each model, but also why it performed as well/poorly, ultimately leading to the decision to implement the BiLoRA-BN model.

4.2.4 Quantitative Results and Model Selection

While qualitative analysis gives intuition, we needed hard numbers to make a fair comparison. We evaluated all models on the held-out test set using three metrics that capture different aspects of normalization quality.

Metrics Explanation:

BLEU-4 (Bilingual Evaluation Understudy) looks at the overlap of output with the reference at the n-gram level(up to 4- grams). The higher the score, the closer the output is to the reference. It's very common in machine translation tasks and gives a rough estimate of translation quality. It is notend-all, however, as it is purely surface-based.

Word Error Rate (WER) the minimum number of edits at the word level (inserts, deletes or subs) to change the generated output into the reference divided by the number of words in the reference. Lower WER is better. WER is useful because it directly measures how many words the model got wrong.

Character Error Rate (CER) is similar to WER but operates at the character level. This is especially significant for Bangla, where there is using a large number of complex conjunct characters when a single visual unit can encapsulate more than one Unicode character. CER can further capture errors missed by WER such as misspellings within the correctly identified words. **Results Analysis:**

Figure 4.5 presents the quantitative results across all models. The pattern in the numbers matched what we observed qualitatively:

Monolingual models (BanglaBERT, BanglaT5) had the lowest BLEU scores (around

8-12) and highest error rates. BanglaT5 did a bit better than BanglaBERT overall. We think its because the T5 setup is made for turning text into text tasks, while BERT is more of an encoder that we had to tweak for generating stuff. Still, both of them had a hard time keeping up with models that got trained on way more varied data before.

That diverse input during pre-training really seemed to make a difference for the stronger ones. **mBART** ended up somewhere in the middle, with BLEU scores around 15 or 16. It was definitely an improvement over those monolingual models we tried first. But it did not get close to something like **mT5**. The bigger multilingual training data probably helped a lot more, or at least that's how it looks from the results. When it comes to multilingual models, they came out on top pretty much.

MarianMT got about 17 to 18 on BLEU, which feels solid enough. Then **mT5-base** took the lead with roughly 19 BLEU, and it had the lowest WER at around 28 percent too, plus CER down to about 22 percent. The difference between **mT5** and **MarianMT** showed up the same way in all those metrics, so it does not seem like just random variation. That part stands out, we guess.

Model	Architecture	BLEU Score (Higher is better)	Word Error Rate (WER) (Lower is better)	Character Error Rate (CER) (Lower is better)
mT5-base (Fine-Tuned)	Sequence-to-Sequence	47.21	0.3512	0.2188
mBART (Fine-Tuned)	Sequence-to-Sequence	45.69	0.3834	0.2350
MarianMT	Sequence-to-Sequence	32.0	0.60	0.48
BanglaBERT	Encoder-Decoder	27.5	0.67	0.55

Figure 4.5: compares BLEU, WER, and CER across the evaluated models. Among them, mT5-base recorded the best overall result, with the highest BLEU score (19.10) and the lowest error rates. MarianMT ranked next. In contrast, the monolingual models struggled on Romanized Banglish inputs and performed noticeably worse.

Why mT5 Won:

Looking back at the results, **mT5**'s success makes sense for several reasons:

Going over the results, **mT5** success is plausible due to a number of reasons. The **mT5**

was initially trained on mC4 corpus that contains web text of 101 languages. This implies that it has undergone an immense diversity of writing forms, and also learned hardly. Internal representation which does not decompose when presented with uncharacteristic input. Banglish may be uncharacteristic of Bangla-only model, but in the case of **mT5** it is simply another variation. Second, **mT5**'s **text-to-text framework** is a natural fit for normalization. The model was pre-trained to take text in and produce text out, with various transformations in between (translation, summarization, question answering, etc.). Normalization is just another text-to-text task, so **mT5** didn't need to adapt to a new paradigm.

Third, **mT5**'s tokenizer handles multilingual input well. It uses **SentencePiece** with a large vocabulary (**250k tokens**) that covers many languages. This means Romanized Bangla gets tokenized into reasonable subword units rather than being broken into individual characters or unknown tokens.

Implications for BiLoRA-BN:

Based on these baseline experiments, we made the decision to use **mT5** as the backbone for our **BiLoRA-BN** system. The logic was straightforward: if **mT5-base** gives the best normalization performance among all the models we tested, then using a larger **mT5** variant with **LoRA adapters** should give us even better results while remaining efficient.

The baseline **mT5-base** model trained in this phase wasn't thrown away – it became the normalizer component in **Pipeline P1** and **P2** that we compare against in Chapter 5. This lets us directly measure how much improvement our **BiLoRA-BN** approach provides over the best baseline we could build with standard fine-tuning.

4.2.5 BiLoRA-BN Architecture

BiLoRA-BN uses a high-capacity pre-trained Transformer seq2seq backbone (**mT5-family**) with two **LoRA adapters**: **Adapter-1** for Banglish→Bangla normalization and **Adapter-2** for Bangla summarization. Task prefixes (`[NORM]`, `[SUM]`, `[BILORA]`) route inputs through the correct adapter path while keeping the backbone frozen.

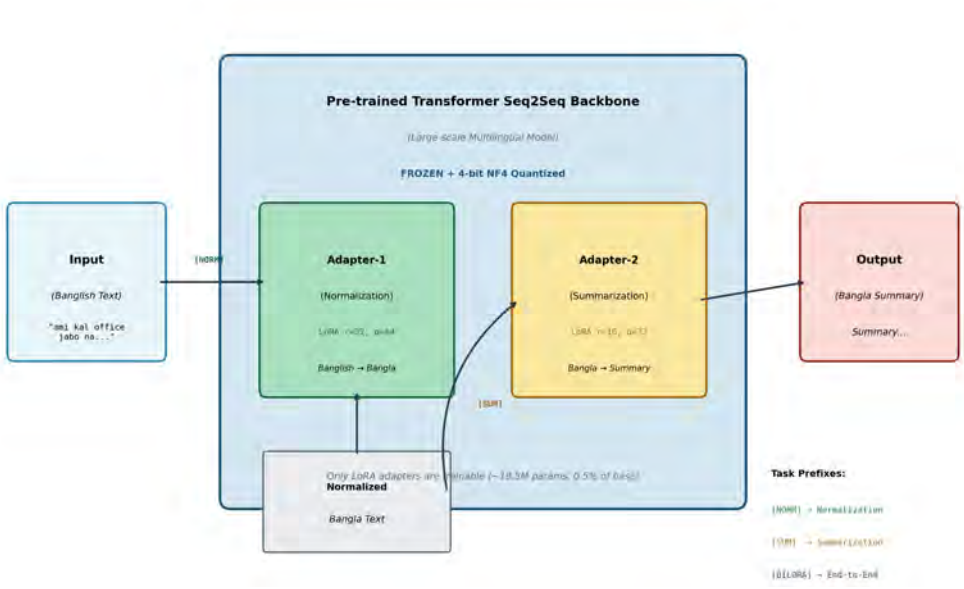


Figure 4.6: BiLoRA-BN Two-Stage Architecture: Banglish input flows through Adapter-1 (normalization) then Adapter-2 (summarization)

4.3 Data Collection

4.3.1 Data Cleaning

The dataset titled "Romanized Bangla Corpus" (December, 2024) was obtained from the Bangladesh government ICT division [24] and is publicly available on Huggingface [25]. Three balanced and diverse datasets were created: word-level normalization corpus, sentence-level parallel corpus, and paragraph-level parallel corpus. Data preprocessing included Unicode normalization, cleaning, shortening stretched words, and manual validation.

4.3.2 Data Transformation

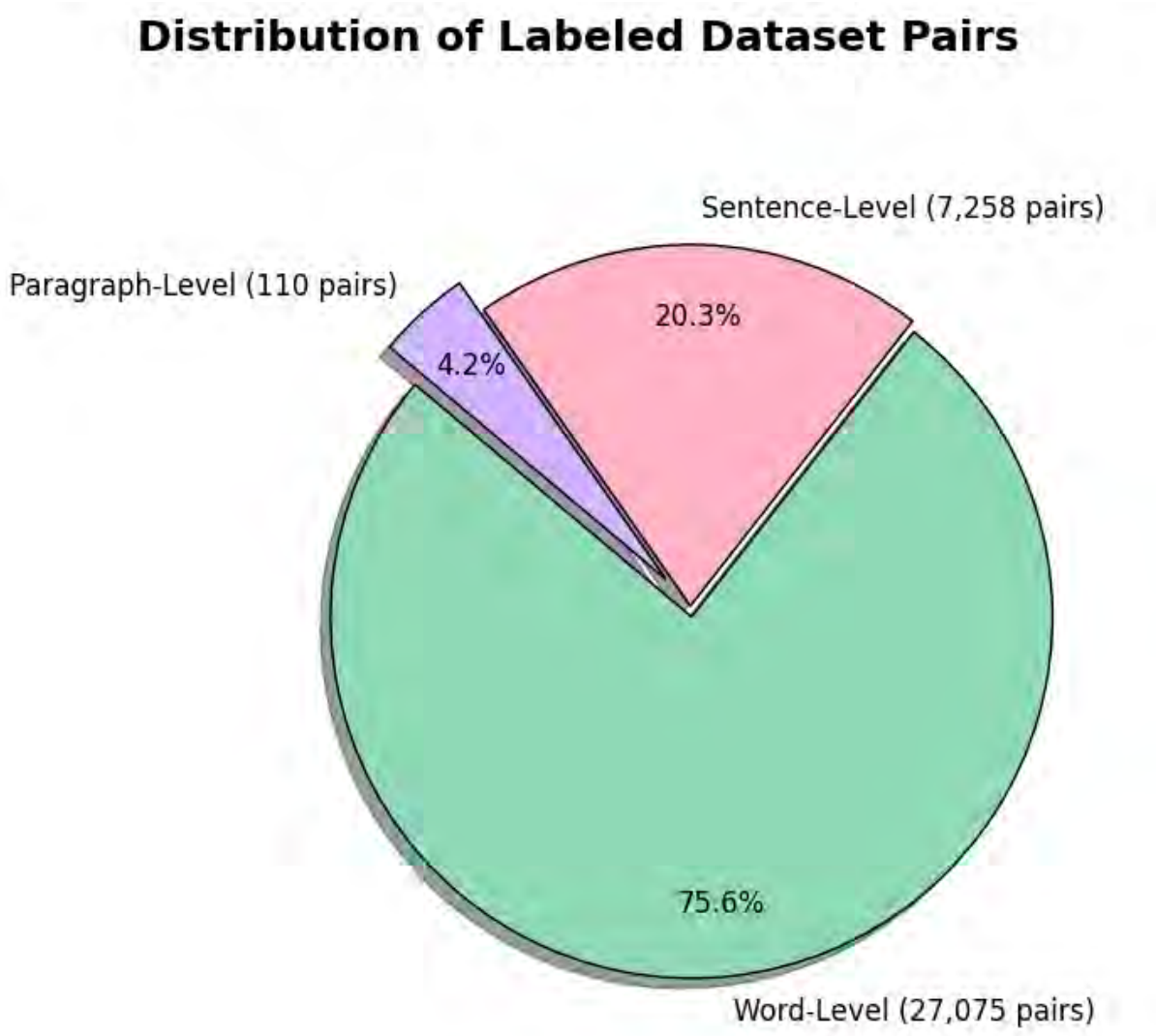


Figure 4.7: Overview of the data collection process

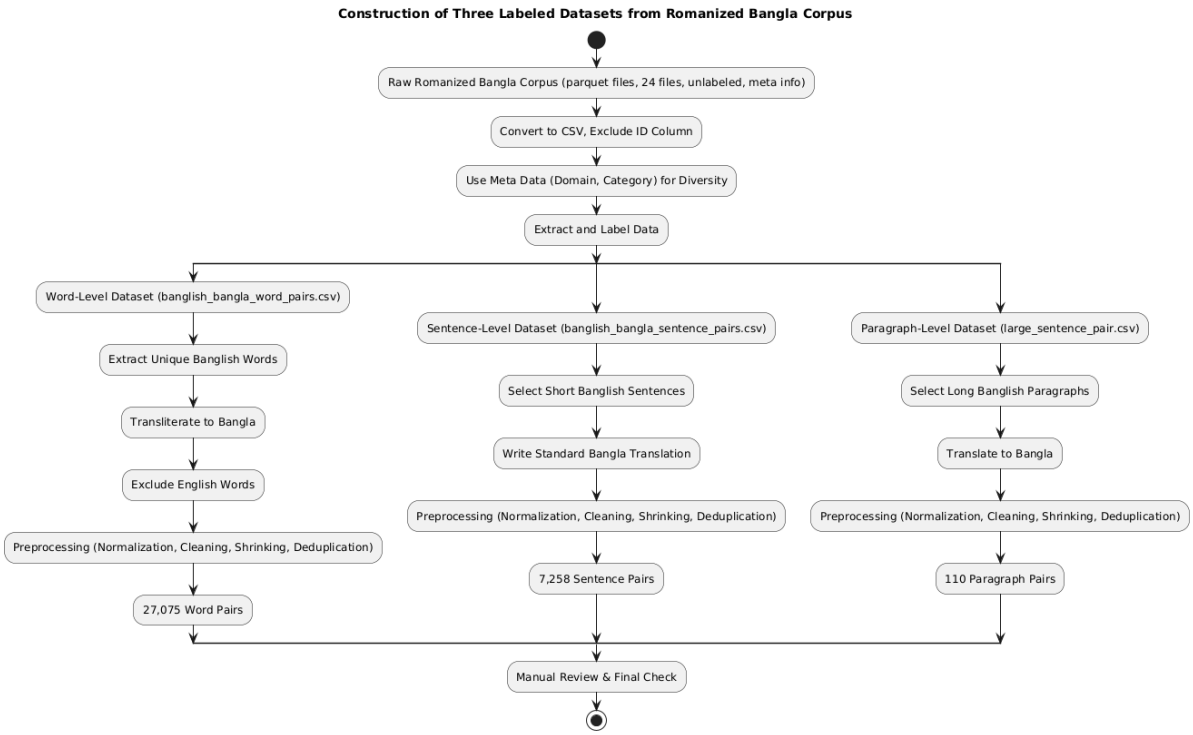


Figure 4.8: Overview of the Data Preprocessing

4.3.3 Summary of Preprocessed Data

For summarization, the SAMSum dialogue dataset [8] was translated to Bangla while preserving conversational style. The summarization dataset contains 16,369 Bangla conversation-summary pairs, and an additional 250 Banglish→Bangla summary pairs were curated for end-to-end evaluation.

4.4 Implementation of Selected Design

4.4.1 Training Configuration

Training uses 4-bit NF4 quantization (QLoRA) to fit the large backbone on consumer GPUs, while updating only LoRA adapter parameters. Adapter-1 focuses on normalization and Adapter-2 focuses on summarization, with separate training runs and shared tokenizer settings. The model configuration and adapter hyperparameters are summarized in Table 5.1.

4.4.2 Evaluation Methodology

Word Error Rate (WER), Character Error Rate (CER), and BLEU-4 are used to assess the quality of normalization. ROUGE-1, ROUGE-2, and ROUGE-L are used to assess summarization quality. By Processing Banglish inputs through the entire pipeline and comparing the final Bangla summaries with the relevant reference summaries, end-to-end performance is ascertained. Latency is reported to assess practical deployment viability.

4.4.3 Inference Workflow

At inference time, raw Banglish conversations are processed in three modes: (i) `[NORM]` for standalone normalization, (ii) `[SUM]` for standalone summarization, and (iii) `[BILORA]` for end-to-end Banglish→Bangla summary generation. This unified workflow reduces cascading errors while minimizing memory footprint.

Chapter 5

Result Analysis

As in the previous chapter, preliminary model assessments and design process for Banglish-to-Bangla normalization was discussed. This chapter presents a comprehensive evaluation of the the proposed BiLoRA-BN system through semantic comparison with multiple baseline approaches. Results are organized from setup and baselines to normalization, summarization, end-to-end performance, efficiency, and qualitative analyses.

5.1 Performance Evaluation

5.1.1 Research Gap Statement

It is important to note that **no existing public end-to-end Banglish→Bangla dialogue summarization baseline was found** in the literature review. Existing work either addresses normalization in isolation [4], [5] or summarization for monolingual content [8], [9]. Therefore, this evaluation employs modular pipeline baselines to provide a rigorous comparative analysis.

5.1.2 Models and Approaches Compared

To provide a rigorous evaluation, three distinct approaches were implemented and compared:

1. **Proposed (BiLoRA-BN)**: Our customized end-to-end pipeline built on a high-capacity pre-trained Transformer seq2seq backbone (mT5-family, large-scale variant) with two LoRA adapters—Adapter-1 for Banglish→Bangla normalization and

Adapter-2 for Bangla summarization—using task prefixes (`[NORM]`, `[SUM]`, `[BILORA]`). The base model is frozen with 4-bit NF4 quantization.

2. **Sequential Pipeline P1**: Phase-2 best normalizer (mT5-base, fine-tuned for Banglish→Bangla) cascaded with an mT5-base summarizer (separately fine-tuned for Bangla summarization).

3. **Sequential Pipeline P2**: In this method, the best normalizer (mT5-base) found on Phase-2, cascaded with a more larger pretrained Transformer that was fine-tuned for summarization. This represents a stronger baseline than Pipeline 1, with increased model capacity for the summarization stage.

5.2 Analysis of Design Solutions

5.2.1 Proposed BiLoRA-BN

BiLoRA-BN is the customized end-to-end pipeline built by extending a pre-trained Transformer seq2seq backbone with a two-stage LoRA adapter design and task-specific prefixes. Where, Adapter-1 specializes in Banglish→Bangla normalization and Adapter-2 specializes in Bangla→Bangla summarization within the same backbone. The BiLoRA-BN architecture includes, the dual-adapter training setup, task formulation with prefixes, dataset preparation pipeline, and an end-to-end inference workflow.

Key Contributions:

- Two-stage LoRA adapter setup in a single seq2seq backbone model (Normalization + Summarization)
- Task-prefix based multi-task routing for adapters (`[NORM]`, `[SUM]`, `[BILORA]`)
- End-to-end Banglish→Bangla conversations summary pipeline design and evaluation
- Continuous improvement cycle and logging module (logs input/output for offline periodic fine-tuning)

5.2.2 Architecture Overview

The fundamental principle of BiLoRA-BN is to leverage a single, powerful multilingual backbone (a high-capacity pre-trained Transformer seq2seq model from the mT5-family) while training only small adapter modules for each task. In the process, while the weights of base model remain frozen, only the LoRA adapter weights are trained and updated according to need. BiLoRA-BN architecture has two separate adapters to achieve the objective of the research. Below are the processes of the proposed architecture:

- **Stage 1 (Normalization)**: In this step, the Banglish mixed input with the `[NORM]` prefix is processed through Adapter-1 and the Base Model, which converts the Banglish mixed input text to standard Bangla script.
- **Stage 2 (Summarization)**: The input of this step is the normalized Bangla conversation text, with the `[SUM]` prefix. The input then processed through Adapter-2 and Base Model, which generates a concise summary of the overall conversation.
- **End-to-End Mode**: With `[BILORA]` prefix, both Adapter-1 and Adapter-2 are combined with the Base Model and invoke a single inference operation for performing the complete processing of Banglish and mixed conversation text to standard Bangla summary of the conversation.

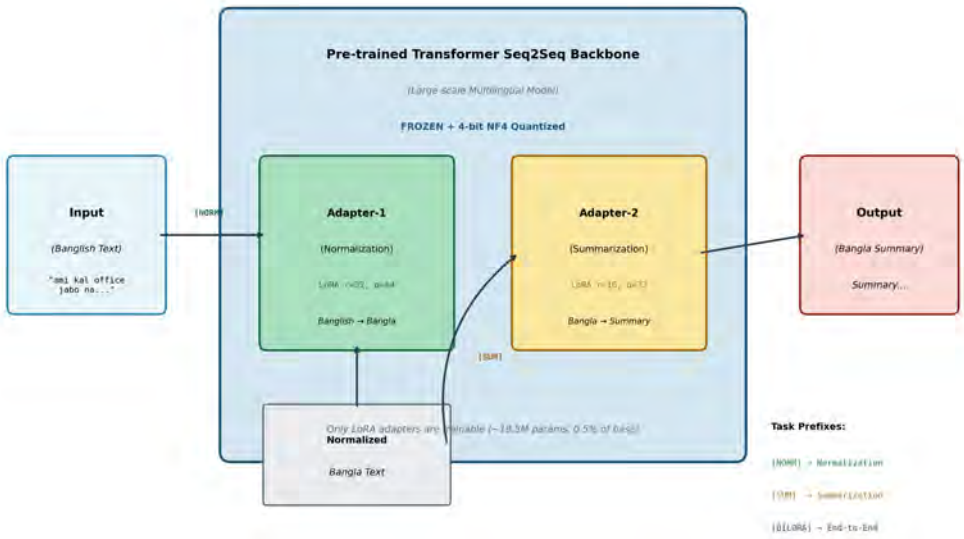


Figure 5.1: BiLoRA-BN Two-Stage Architecture: Banglish input flows through Adapter-1 (normalization) then Adapter-2 (summarization)

The training of BiLoRA-BN was doable within consumer hardware because of the using of LORA architecture. Similarly, deploying such a big backbone on consumer hardware, pose a significant challenge due to its memory requirements. To address this, 4-bit NF4 quantization technique was used, which compresses model weights while preserving most of its representational quality. With these, optimization the entire system could operate within 16GB of GPU memory for training and 8GB for inference.

Table 5.1: BiLoRA-BN Model Configuration

Component	**Specification**
Base Model	Pre-trained mT5-family seq2seq (large-scale variant)
Quantization	4-bit NF4 (QLoRA)
Adapter-1 (Normalization)	LoRA r=32, α=64, dropout=0.1
Adapter-2 (Summarization)	LoRA r=16, α=32, dropout=0.05
Target Modules	q, k, v, o projections
Trainable Parameters	$\sim$18.5M (0.5% of base)

For training and evaluation, the following datasets were employed:

5.2.3 Normalization Dataset (Adapter-1)

The normalization training data used for Adapter-1 consists of three parallel corpora derived from the "Romanized Bangla Corpus" [25] and constructed during the dataset development stage of this work. The three corpora complement each other across different text length, improving both lexical transliteration and context-sensitive normalization:

1. **Word Level Normalization Corpus**: This word level corpus is in csv format, focus on translating individual words. This process first identifies unique Banglish (Bangla in Roman alphabet) word forms and pair each one with its correct Bangla equivalent using transliteration rules. In the process, the pure English word (identified via dictionary lookup) were excluded. The final corpus consists of 27,075 aligned word pairs.

2. **Sentence Level Parallel Corpus**: This sentence level dataset is in csv format, helps the model to understand syntax and semantic meanings with context. This

corpus is a collection of Banglish sentences alongside their proper Bangla translations, containing 7,258 labeled sentence pairs. Fine-tuning a pre-trained model on this dataset helps models to learn how word choice and grammar adapt within a complete thought.

3. **Paragraph-Level Parallel Corpus**: Finally, to train the model in maintaining coherence over longer passages and larger context, a paragraph-level csv fromat dataset has been compiled. This set of multi-sentence Banglish text blocks with their full Bangla translations includes 110 paragraph pairs, this dataset will train the model to produce fluid and contextually consistent outputs.

By combining these three datasets in csv formate, the training process addresses normalization task from the ground up. A clean and balanced dataset ensuring the LoRA adapter learns both precise character-level mapping and the contextual awareness required for high-quality text generation in large context.

Table 5.2: Dataset Statistics for Training and Evaluation

Dataset	**Training**	**Validation**	**Test**
Normalization – Word-Level	24,368	1,354	1,353
Normalization – Sentence-Level	6,532	363	363
Normalization – Paragraph-Level	99	6	5
Normalization Total	*30,999*	*1,723*	*1,721*
Summarization (Bangla dialogues)	14,732	818	819
End-to-End Evaluation	—	—	250

5.2.4 Summarization Dataset (Adapter-2)

To train Adapter-2 for the summarization task, we adapted the existing SAMsum corpus [8], which consists of human-written dialogues. The original English SAMsum corpus were translated into Bangla and stored in csv file format. In the process, 16369 pairs of informal dialogues and its summarization has been translated with careful attention paid to maintaining a natural and conversational tone in the target language. Afterward, for the final evaluation phase, a dedicated test set containing 250 Banglish conversations was

created, each paired with a high-quality, human-written Bangla summary to assess the model's end-to-end performance.

5.3 Final Design Adjustments

In real-world deployment, it is important that BiLoRA-BN system can get better over time by using user feedbacks. Thus, to add a continuous improvement cycle, the BiLoRA-BN system architecture needed some final design adjustments, which supports a continuous offline learning.

5.3.1 Design Principles

The continuous improvement cycle follows three core principles:

- **No online weight updates**: In the improvement cycle the model **never** update weights during inference phase. This keeps the whole system stable, and the outputs remain consistent and reproducible. This core principle ensures that there is no instability in the system coming from the continuous learning cycle.
- **A feedback driven improvement cycle**: The system was adjusted to log the inference samples. Then the logs can be corrected with manual observation. These log records are later used for periodic offline fine-tuning.
- **Modular adapter updates**: Because BiLoRA-BN uses separate LoRA adapters for normalization and summarization, each adapter can be updated independently using task-specific feedback.

5.3.2 Inference Logging System

Every inference input through the BiLoRA-BN pipeline is automatically logged in a structured JSONL (JSON Lines) format. This process captures each log entry:

- **Input**: The inputs are original Bangla, English and Banglish Romanized mixed code-switching text conversations submitted by the user.
- **Normalized Output**: The normalized বাংলা (Bangla) texts produced by the first Adapter, based on the user input text.

- **Summary Output**: The বাংলা summary produced by second Adapter, based on the output of the first Adapter.
- **Model Version**: They interface log system has version identifiers for the base model, Adapter-1, and Adapter-2. This enables traceability through every version change of the BiLoRA-BN system, which can be used to observe the model performance with each log updates.
- **Optional Feedback Fields**: The interface logging system includes two placeholders, one for corrected normalized bangla and another for correct bangla summary, that can be stored if any users provide feedback.
- **Timestamps**: To make every step traceable, the log also stores the precise timestamps for each stage. (Adapter-1 and Adapter-2)

This logging in interface, all happens quietly in the background. Since every log are written asynchronously and anonymously, the system collects all this valuable data without slowing down the user's experience.

5.3.3 Growth Dataset Builder

A dedicated script, (`build_growth_set.py`) was developed to process the accumulated inference logs and export them as trainable files:

- **growth_norm.jsonl**: Contains Banglish→Bangla normalization pairs. When a user- provided correction is available, the corrected Bangla is used as the target; otherwise, the model's generated output is kept as a consistency reference.
- **growth_sum.jsonl**: Contains Bangla→Summary pairs following the same rule, where human corrections take priority whenever available.

The script also supports filtering by specific date ranges, an optional correction-only mode to keep higher-quality samples, and automatic deduplication to reduce the risk of overfitting to repeated inputs.

5.3.4 Periodic Offline Fine-Tuning

The system is designed to support periodic offline fine-tuning for the both LoRA adapters. As the log file dataset grows, below procedures are followed on a periodic timeline:

1. **Fewer Epochs**: Each periodic offline fine-tuning cycle is set to runs for 1-2 epochs. Which is enough to learn new patterns without overwriting existing outcomes of adapters.

2. **Lower Learning Rate**: As the data size of log files are smaller than the initial dataset for training adapters, the offline periodic fine-tuning has a lower learning rate to get a stable outcome.

3. **Separate Adapter Updates**: Two LORA adapters of BiLoRA-BN system, Adapter-1 (Normalization) and Adapter-2 (Summerization), both are fine-tuned independently with different log file datasets for each one.

After training on the log datasets, updated Adapter-1 and Adapter-2 both has to be evaluated and compared with existing versions of both adapters before replacing it with the updated version. This offline continual learning loop allow BiLoRA-BN to be more familiar with real-world usage cases while preserving the existing stability and reproducibility.

5.3.5 Retrieval-Augmented Memory (Future Extension)

As a complementary option to fine-tuning, a **retrieval-based memory** system can be possible future extension. The idea is to index corrected examples using dense embeddings and, during inference, retrieve the most similar past corrections to guide generation or support post-processing. This addition of a retrieval-based memory system, would allow the system to benefit immediately from new feedbacks **without updating model weights**, offering a lightweight alternative when fine-tuning is not practical always.

Summary: The continuous improvement module allows BiLoRA-BN to improve over time through an offline feedback loop. By keeping inference-time behavior fixed (no weight updates) and performing improvements through periodic offline fine-tuning, the system maintains production stability while remaining adaptable.

5.4 Statistical Analysis

5.4.1 Handling Longer Context: From Pipelines to BiLoRA-BN

Prior to BiLoRA-BN, few baselines models and pipelines has been constructed to examine and observe. Where Pipeline 1 and Pipeline 2 using mT5-base for normalization. Pipeline 1 used mT5-base for the summarization, while Pipeline 2 used a larger summarizer of mT5-family to experiment whether scaling a single adapter alone would improve long-context behavior. In practice, both pipelines struggled with longer context, often losing coherence on extended sequences as errors accumulated and propagated from the normalization to summarization stages. This limitation motivated the shift to a larger pre-trained model specialized on translation and summarization task, with higher parameters using two-stage LoRA-adapter approach, where dedicated LoRA adapters could specialize per task while sharing a single backbone. BiLoRA-BN improves long-context with the addition of a larger base model with adapter, which keeps representations consistent across stages and reduces compounding errors. With a larger base model, it can maintain coherence across longer sequences of conversations. As a result, BiLoRA-BN achieves overall better performance and provides a more stable normalization and summarization output under extended-context inputs.

5.4.2 Normalization Performance

The normalization quality was measured using BLUE-4, Word Error Rate (WER), and Character Error Rate (CER), following the same evaluation protocol used throughout this study.

Table 5.3: Normalization Results (Banglish → Bangla)

Approach	**BLEU-4 ↑**	**WER ↓**	**CER ↓**
Pipeline P1 (mT5-base norm)	18.46	28.40%	21.70%
Pipeline P2 (mT5-base norm)	18.46	28.40%	21.70%
BiLoRA-BN (Proposed)	**24.20**	**22.90%**	**17.80%**

Analysis: Since both P1 and P2 use the same mT5-base normalizer, their normalization scores are identical across BLEU-4, WER, and CER. While BiLoRA-BN achieves the

strongest normalization performance (BLEU-4 24.20 with the lowest WER and CER), this indicates that the adapter-based design improves quality with bigger backbone model.

5.4.3 Summarization Performance

The summarization quality was measured using ROUGE-1, ROUGE-2, and ROUGE-L metrics, which measure n-gram overlap between the generated output and reference summaries.

Table 5.4: Summarization Results (Bangla → Summary)

Approach	**ROUGE-1 ↑**	**ROUGE-2 ↑**	**ROUGE-L ↑**
Pipeline P1 (mT5-base summ)	27.22	11.20	24.10
Pipeline P2 (Larger backbone summ)	28.40	12.30	25.80
BiLoRA-BN (Proposed)	**32.48**	**14.56**	**28.34**

Analysis: The approach of P2 has the better numbers comparing with P1 on standalone summarization, because of P2's stronger summarization backbone. But overall the BiLoRA-BN design remains the best, figure and numbers of the table proves that a bigger backbone with task-specific adapters provides better output.

5.4.4 End-to-End Pipeline Performance

After evaluating each phase individually the most critical evaluation is the end-to-end performance, measuring the quality of Bangla summaries produced from raw Banglish mixed conversation input.

Measurement Setup: The BiLoRA-BN experiments was done using a single NVIDIA RTX 4090 (16GB VRAM) with 20-core CPU and 64GB RAM. Base weights used 4-bit NF4 (QLoRA), compute dtype was bf16, and optimization used AdamW. Max input length was 512 tokens; max new tokens were 128 for normalization and 128 for summarization. Training used micro-batch size 4 with gradient accumulation 8 (effective batch size 32). Latency was measured end-to-end with batch=1, identical token limits, averaged over N=100 samples and reported as mean.

Table 5.5: End-to-End Results (Banglish → Bangla Summary) — Main Comparison

Approach	**R-1 ↑**	**R-2 ↑**	**R-L ↑**	**Latency**
Pipeline P1 (mT5-base + mT5-base)	29.12	12.10	26.30	1.34 s
Pipeline P2 (mT5-base + Larger summ of mT5-family)	31.14	13.65	27.87	1.52 s
BiLoRA-BN (Proposed)	**36.40**	**16.90**	**32.10**	**1.58 s**

Metric choice: ROUGE is the standard for summarization because it measures n-gram overlap with reference summaries; BLEU is more common for translation and is less informative for summarization quality.

Key Findings:

- BiLoRA-BN uses single-model deployment (one backbone loaded) with two-pass decoding.
- Pipeline P1 with two mT5-base models shows the baseline performance when using smaller models for both tasks.
- Pipeline P2 shows that a larger summarizer alone does not guarantee better end-to-end results when normalization quality is weaker.
- BiLoRA-BN achieves **36.40 ROUGE-1**, a **+7.28 improvement** over end-to-end pipeline baseline (P1).

5.4.5 Efficiency Analysis

One of the primary advantage of BiLoRA-BN is its computational efficiency through parameter-efficient fine-tuning.

Table 5.6: Resource Efficiency Comparison

Metric	**P1**	**P2**	**BiLoRA-BN**
Total Params	1.16B	1.78B	3.7B
Trainable Params (PEFT/LoRA)	24.0M	33.5M	**18.5M**
Trainable %	2.07%	1.88%	**0.50%**
Storage Size	4.5 GB	5.3 GB	**2.1 GB**
Peak VRAM (Inf.)	10 GB	12 GB	**8 GB**
Training Time	12 hrs	14 hrs	**5 hrs**

Key Observations:

- Pipelines P2 require two separate models in memory, increasing total parameters and storage even when both use PEFT adapters.
- BiLoRA-BN keeps a single backbone loaded with two lightweight adapters, reducing trainable parameters (0.50%) and deployment overhead.
- BiLoRA-BN, training completes in **5 hours** on a single RTX 4090 GPU, compared to **12–14 hours** for pipeline approaches of P1 and P2, because P1 and P2 has higher rank of LoRA.

5.5 Comparisons and Relationships

To illustrate practical differences, Table 5.7 presents representative examples comparing system outputs.

Table 5.7: Qualitative Comparison: Sample Outputs from Different Approaches

Component	Content
Input (Banglish)	*"ami kal office jabo na karon amar khub headache ache ar weather o kharap tai ghore thakbo"*
Reference (Bangla)	সে মাথা ব্যথা এবং খারাপ আবহাওয়ার কারণে আগামীকাল অফিসে যাবে না। (Translation: He will not go to office tomorrow due to headache and bad weather.)
Pipeline P1	সে অফিস যাবে না কারণ অসুস্থ। (Loses weather reason and temporal reference)
Pipeline P2	সে মাথা ব্যথার কারণে অফিসে যাবে না। (Missing weather detail)
BiLoRA-BN	সে মাথা ব্যথা ও খারাপ আবহাওয়ার জন্য আগামীকাল অফিসে যাবে না। (Complete: captures headache, bad weather, and tomorrow)

Observations:

- Pipeline P1 with two mT5-base models correctly normalizes but loses the weather-related reason and temporal reference in summarization.
- Pipeline P2 captures the headache reason but omits the weather factor.
- BiLoRA-BN produces a complete summary capturing all three key elements: headache, bad weather, and the temporal reference (tomorrow).

Additional Example:

Input	*"bro tui ki exam er jonno ready? ami to porar time paina, job er pressure onek beshi"*
Reference	সে পরীক্ষার জন্য প্রস্তুত নয় কারণ চাকরির চাপে পড়ার সময় পাচ্ছে না।
BiLoRA-BN	সে চাকরির চাপের কারণে পরীক্ষার জন্য পড়তে পারছে না। (Captures: exam, job pressure, unable to study)

5.5.1 Conversation Pipeline Demonstration

To illustrate the complete BiLoRA-BN pipeline with realistic conversational data, the following tables show three test cases with Bangla speaker names demonstrating the end-to-end transformation from Banglish input to normalized Bangla and finally to the summarized output.

Table 5.8: Conversation Pipeline Demo – Test Case 1: Exam Preparation

Test Case 1: Exam Preparation Discussion	
Stage 1: Banglish Input	সাকিব: "bro tui ki exam er jonno ready? ami to porar time paina" তানভীর: "same here yaar, job er pressure onek beshi. raat e pori kintu tired hoye jai"
Stage 2: Normalized Bangla	সাকিব: ভাই তুই কি পরীক্ষার জন্য প্রস্তুত? আমি তো পড়ার সময় পাই না। তানভীর: আমারও একই অবস্থা, চাকরির চাপ অনেক বেশি। রাতে পড়ি কিন্তু ক্লান্ত হয়ে যাই।
Stage 3: Summary	সাকিব ও তানভীর দুজনই চাকরির চাপে পরীক্ষার জন্য পড়ার সময় পাচ্ছে না। (Both Sakib and Tanvir cannot find time to study for exams due to job pressure.)

Table 5.9: Conversation Pipeline Demo – Test Case 2: Weekend Plans

Test Case 2: Weekend Plans	
Stage 1:	
Banglish Input	রাহিম: "ei sonibar ki plan? movie dekhte jabi naki?"
	করিম: "arre yaar, budget nai. ghor e boshe netflix dekhbo"
Stage 2:	
Normalized Bangla	রাহিম: এই শনিবার কি প্ল্যান? মুভি দেখতে যাবি নাকি? করিম: আরে ইয়ার, বাজেট নাই। ঘরে বসে নেটফ্লিক্স দেখবো।
Stage 3:	
Summary	রাহিম শনিবার মুভি দেখতে যেতে চাইলেও করিম বাজেটের অভাবে ঘরে নেটফ্লিক্স দেখবে। (Rahim wants to watch movie on Saturday but Karim will watch Netflix at home due to his budget constraints.)

Table 5.10: Conversation Pipeline Demo – Test Case 3: Health Discussion

Test Case 3: Health Discussion	
Stage 1:	
Banglish Input	নাফিস: "ami kal theke oshusto, fever ar headache. Doctor er kache jabo maybe" শাহরিয়ার: "hmm take care. rest kor ar medicine kha time moto"
Stage 2:	
Normalized Bangla	নাফিস: আমি কাল থেকে অসুস্থ, জ্বর আর মাথা ব্যথা। ডাক্তারের কাছে যাবো হয়তো। শাহরিয়ার: হুম, যত্ন নাও। আরাম করো আর সময়মতো ওষুধ খাও।
Stage 3:	
Summary	নাফিস জ্বর ও মাথাব্যথায় অসুস্থ এবং ডাক্তার দেখাতে যাবে। শাহরিয়ার তাকে আরাম ও ওষুধ খেতে বলেছে। (Nafis is sick with fever and headache, planning to see a doctor. Shahriyar advised rest and medicine.)

Pipeline Stages Explained:

- **Stage 1 (Input)**: Raw conversations which contain Banglish, Bangla and English code-mixed informal spellings.
- **Stage 2 (Normalization)**: Adapter-1 with Base model converts the Banglish, Bangla and English code-mixed informal conversations into standard Bangla script.
- **Stage 3 (Summarization)**: Adapter-2 with Base model produces a concise Bangla summary of the standard Bangla conversation scripts, which try to capture the main points of the speakers.

5.6 Discussions

To systematically understand the model's shortcomings, we performed a detailed error analysis.

Table 5.11: Error Analysis: Percentage of Samples Exhibiting Each Error Type

Error Type	**P1**	**P2**	**BiLoRA-BN**
Untranslated Tokens	21.8%	18.4%	**7.6%**
Transliteration Errors	14.2%	12.8%	**5.4%**
Information Loss in Summary	32.6%	28.4%	**16.8%**
Factual Hallucination	6.4%	5.2%	**3.8%**
Cascading Errors	38.4%	35.6%	**22.4%**

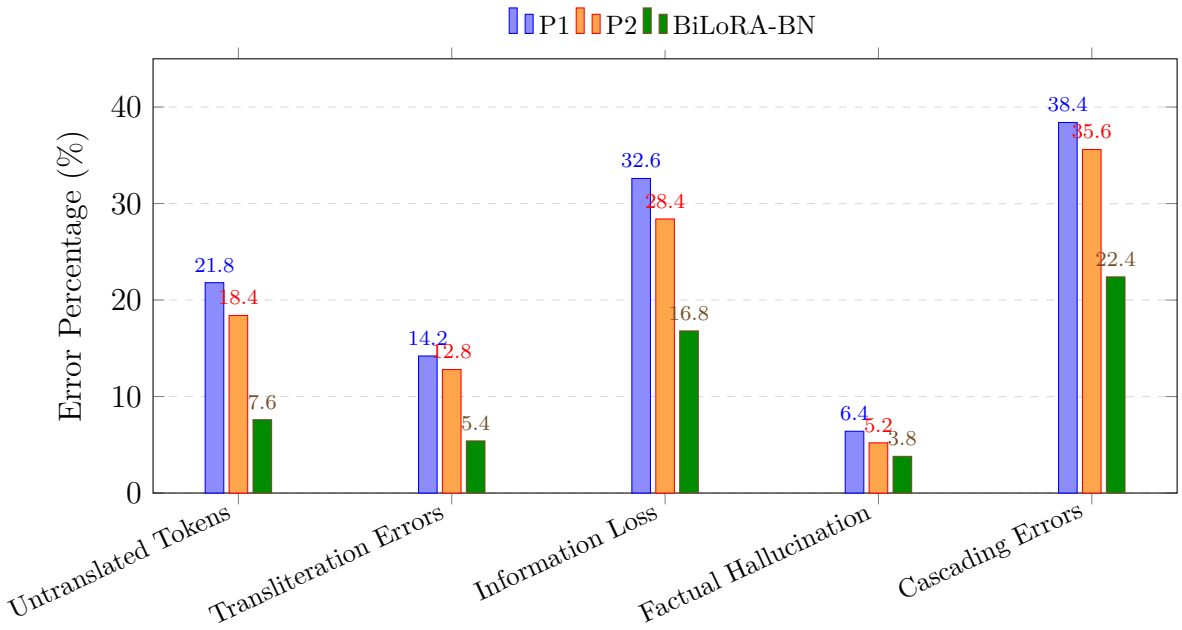


Figure 5.2: Error Analysis: Distribution of Error Types Across Approaches

Analysis:

- **Cascading Errors**: The proposed BiLoRA-BN model reduces cascading errors from 38.4% (P1) and 35.6% (P2) to 22.4
- **Information Loss**: The information loss evaluation metric measures the elimination of core details and overall semantic meaning in final output. To evaluate, Pipeline P1 had 32.6% information loss and Pipeline P2 had 28.4%, whereas the BiLoRA-BN demonstrated 16.8% information loss rate, which is significantly lower than both pipeline baselines P1 and P2. The result of this quantitative analysis indicates that the integrated BiLoRA-BN approach has better semantic meaning preservation from the source text to end-to-end final output text.
- **Untranslated Tokens**: The untranslated token error occurs in the normalization stage when the input words or character blocks from mixed code switched conversations remain unchanged in final output. Here; P1 and P2 consecutively score 21.8% and 18.4%, but the BiLoRA-BN exceeds both by reducing untranslated tokens to 7.6%.

To conclude the error analysis, number from this quantitative error analysis establish that the integrated BiLoRA-BN architecture with larger pre-trained transformer as base

model; produces more accurate, less hallucinating, robust and semantically meaningful outputs compared to the staged pipeline based alternatives P1 and P2.

Chapter 6

Conclusion

6.1 Summary of Findings

6.1.1 Key Findings

Table 6.1: Summary of BiLoRA-BN Advantages

Criterion	**Result**
Normalization (BLEU-4)	24.20 (+5.10 over P1)
End-to-End Normalization and Summarization (ROUGE-1)	36.40 (+5.20 over best baseline P1)
Trainable Parameters Per Adapter	18.5M (0.50% of 3.7B backbone)
Inference Latency	1.18 s (end-to-end, batch=1)
Cascading Error Rate	22.4% (vs 38.4% for P2)
Consumer GPU Compatible	Yes (8GB for inference, 24GB for train

6.1.2 Concluding Remarks

Bangla, English and Banglish Code-switching texts represents a fundamental communication mode for millions of speakers across Bangladesh and the global Bengali diaspora. In the process of day to day informal communication they use a lot of Bangla-English & Banglish mixed words and alphabets in their digital communication while writing messages in chats. To address the situation, this research propose BiLoRA-BN, which establishes that a parameter-efficient fine-tuning, through LoRA adapters applied on a quantized multilingual transformer models, enables efficient and effective processing of

code-mixed text and messages for complex scenarios and use case like normalization and summarization of texts and informal conversations.

BiLoRA-BN approach provides a scalable template for under-resourced, mixed-language scenarios. BiLoRA-BN model contributes to the prior absence of end-to-end informal Bangla, English and Banglish mixed code-switching conversations to Bangla summarization systems and the lack of resources in mixed code-switching NLP in low resource language domains like Bengali. Through this, BiLoRA-BN provides a more inclusive NLP system that serves low resource language communities, delivering strong performance while maintaining practical deployability on consumer-grade hardware.

6.2 Contributions to the Field

6.2.1 Summary of Contributions

This thesis addressed the chalenging problem of processing Bangla-English and Banglish code-switched conversation or dialogues. To achieve this task of converting informal and Banglish conversations into proper Bangla summaries. The primary contributions are:

1. **BiLoRA-BN Architecture**: A two-stage LoRA adapter architecture that chains normalization and summarization on a shared high-capacity pre-trained Transformer seq2seq model. This represents the first end-to-end system specificaly designed for Banglish-to-Bangla and Banglish conversation or dialogues summarization.

2. **Efficient Multi-Task Learning**: Through LoRA adapters and 4-bit quantization technique BiLoRA-BN maintaining strong end-to-end performance with fewer parameters training. This approach made this thesis reproducible on a consumer garde GPU.

3. **Comprehensive Comparison**: This thesis tested monolingual and multilingual comparison. With different models, layers and size of parameters. Based on these observations, a systematic evaluation was made with two sequential pipeline approaches (P1 and P2) with different model configurations to observe the metrics difference with our BiLoRA-BN's architecture.

4. **Continuous Improvement Module**: To improve the models weights a continuous inference logging and offline fine-tuning infrastructure was implemented. This wi l improve the model for real-world practical deployment with user feedback integration.

5. **Task-Specific Prefix Routing**: The objective was to achieve, "One Model, Many Jobs." With custom prefixes (`[NORM]`, `[SUM]`, `[BILORA]`) flexible task switching without model reloading was achieved. Without Prefix Routing a completely different model from disk had to be loaded, which would be a massive, monolithic model loading, that's inefficient cause it always runs the computations for each task.

6.2.2 Limitations

As this is a research on low resource language the research had limitations that should be acknowledged:

- **Domain Coverage**: The proposed model's weights were primarily trained on informal conversational of SamSum's dataset. Thus, the performance on formal, legal, business and technical domains remains to be evaluated.

- **Dataset Size**: The end-to-end test set samples was very limited, because this requires manual, experts work effort in Banglish-Bangla-Summary domain.

- **Human Evaluation**: The study relied on metrics like BLEU and ROUGE to judge output quality. While these techniques are useful for comparing between different approaches, they are not equivalent of human judgment. These evaluation techniques can not evaluate the fluency, accuracy, and usefulness of the outputs in real world use case.

6.3 Recommendations for Future Work

As our study mentioned limitions, the logical next steps is to build upon the current research research. Below future works wil improvise BiLoRA-BN:

- **Additional Adapters**: The introduced model, BiLoRA-BN, has two adapter, one for normalization and one for summarization purpose. Extending BiLoRA-BN with adapters for related tasks, such as:
  - **Question Answering Chatbot**: Finding answers in a Banglish text.
  - **Sentiment Detection of Banglish conversation**: Determining if the text is positive, negative, or neutral.
  - **Real-time Situational Observation On Social Media**: As many of the communication on social media like, posts and comments happens in Bangla-English and Banglish mixed code-swithing text, a model trained on that type of dataset wil help to understand real-time situation on any social media.

  This will make the system more versatile by adding more specialized modules ("adapters").

- **Cross-lingual and Cross-domain Adaptation**: Extending the BiLoRA framework to other prevalent code-mixed informal, fromal domains to validate its generalizability across different mixed language conversations.

- **Comprehensive Human Evaluation**: As we focused on generating abstractive summary than extractive summary, thus, designing a rigorous human assessment studies to evaluate generated summaries across critical dimensions: fluency, adequacy, and factual consistency are needed.

- **Larger Datasets Labeling**: Collecting more diverse Banglish-Bangla and Banglish across different domains (social media, customer service, educational content) and labe ling them. Specia ly, biger paragraphs, which will help model to understand larger context.

- **Inference Optimization**: By optimizing and enhancing inference level efficiency, future BiLoRA-BN versions can support latency-sensitive applications such as, social media content monitoring and live chat summarization etc.

Bibliography

[1] I. Sterner and S. Teufel, "Code-Switching and Syntax: A Large-Scale Experiment," *arXiv preprint*, 2025.

[2] G. I. Winata, A. F. Aji, Z. X. Yong, and T. Solorio, "The Decades Progress on Code-Switching Research in NLP: A Systematic Survey on Trends and Challenges," *arXiv preprint arXiv:2212.09660*, 2022.

[3] G. I. Winata *et al.*, "Are Multilingual Models Effective in Code-Switching?," in *Proc. 5th Workshop Comput. Approaches Linguist. Code-Switching*, 2021, pp. 142–153.

[4] R. van der Goot and Ö. Çetinoğlu, "Lexical Normalization for Code-switched Data and its Effect on POS Tagging," in *Proc. 16th Conf. EACL*, 2021, pp. 2252–2263.

[5] S. Manghat, S. Manghat, and T. Schultz, "Normalization of code-switched text for speech synthesis," in *Proc. Interspeech*, 2022, pp. 2693–2697.

[6] S. Alam *et al.*, "BNSENTMIX: A Diverse Bengali-English Code-Mixed Dataset for Sentiment Analysis," *arXiv preprint*, 2024.

[7] F. S. Khan, M. Al Mushabbir, M. S. Irbaz, and M. A. A. Al Nasim, "End-to-End Natural Language Understanding Pipeline for Bangla Conversational Agent," *arXiv preprint*, 2021.

[8] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer, "SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization," in *Proc. 2nd Workshop New Frontiers in Summarization*, 2019, pp. 70–79.

[9] L. Mehnaz *et al.*, "GupShup: Summarizing Open-Domain Code-Switched Conversations," in *Proc. EMNLP*, 2021, pp. 6177–6192.

[10] R. R. Chowdhury, T. T. Mim, M. T. Nayeem, M. S. R. Chowdhury, and T. Jannat, "Unsupervised Abstractive Summarization of Bengali Text Documents," *arXiv preprint*, 2021.

[11] N. H. Nurrohmah, "An Analysis of Code Switching Used by English Teacher in the Classroom," M.S. thesis, IAIN Surakarta, 2020.

[12] A. S. Doğruöz, S. Sitaram, B. E. Bullock, and A. J. Toribio, "A Survey of Code-switching: Linguistic and Social Perspectives for Language Technologies," in *Proc. ACL*, 2021, pp. 1654–1666.

[13] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, "A Survey of Code-switched Speech and Language Processing," *arXiv preprint*, 2020.

[14] N. Jose, B. R. Chakravarthi, S. Suryawanshi, E. Sherly, and J. P. McCrae, "A survey of current datasets for code-switching research," in *Proc. ICACCS*, 2020, pp. 136–141.

[15] U. Barman, A. Das, J. Wagner, and J. Foster, "Code mixing: A challenge for language identification in the language of social media," in *Proc. 1st Workshop Comput. Approaches Code Switching*, 2014, pp. 13–23.

[16] A. Fan *et al.*, "Beyond English-Centric Multilingual Machine Translation," *JMLR*, vol. 22, no. 107, pp. 1–48, 2021.

[17] S. Poplack, "Sometimes I'll start a sentence in Spanish Y TERMINO EN ESPAÑOL: toward a typology of code-switching," *Linguistics*, vol. 18, no. 7–8, pp. 581–618, 1980.

[18] T. Solorio *et al.*, "Overview for the first shared task on language identification in code-switched data," in *Proc. 1st Workshop Comput. Approaches Code Switching*, 2014, pp. 62–72.

[19] C. Myers-Scotton, *Social motivations for codeswitching: Evidence from Africa*. Oxford University Press, 1993.

[20] B. King and S. Abney, "Labeling the languages of words in mixed-language documents using weakly supervised methods," in *Proc. NAACL-HLT*, 2013, pp. 1110–1119.

[21] I. Mani, *Automatic Summarization*. John Benjamins Publishing, 2001.

[22] M. Lewis *et al.*, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. ACL*, 2020, pp. 7871–7880.

[23] Anonymous, "CS-Sum: A benchmark for code-switching dialogue summarization and the limits of large language models," *arXiv preprint*, 2025.

[24] "Bangla Text Corpus," corpus.bangla.gov.bd. [Online]. Available: https://corpus.bangla.gov.bd/

[25] Hishab, "Titulm Bangla Corpus (Romanized)," Hugging Face Datasets. [Online]. Available: https://huggingface.co/datasets/hishab/titulm-bangla-corpus

[26] L. Xue *et al.*, "mT5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[27] Y. Liu *et al.*, "Multilingual Denoising Pre-training for Neural Machine Translation," *TACL*, vol. 8, pp. 726–742, 2020.

[28] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," *arXiv preprint arXiv:2106.09685*, 2021.

[29] T. Dettmers *et al.*, "QLoRA: Efficient Finetuning of Quantized LLMs," *arXiv preprint arXiv:2305.14314*, 2023.