

BanglaFeel: Enhanced Bangla Back-Transliteration

by

Masfiq Mahmud (212014007)
Sumaiya Akter Iqra (211014004)
Muhammad Anas (212014020)
Rafit Ahmed Mullick (212014038)
Anika Tasnim Bushra (211014019)

*Capstone project report (CSE 4098C) submitted in partial fulfillment of the
requirements for the degree of*

Bachelor of Science in Computer Science and Engineering

Under the supervision of

Md. Aktaruzzaman Pramanik



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF LIBERAL ARTS BANGLADESH**

SPRING 2025

© *Masfiq Mahmud, Sumaiya Akter Igra, Muhammad Anas, Rafit Ahmed Mullick,*
and *Anika Tasnim Bushra*
All rights reserved

DECLARATION

Project Title	BanglaFeel: Enhanced Bangla Back-Transliteration
Authors	<i>Masfiq Mahmud, Sumaiya Akter Iqra, Muhammad Anas, Rafit Ahmed Mullick, and Anika Tasnim Bushra</i>
Student IDs	212014007, 211014004, 212014020, 212014038, and 211014019
Supervisor	Md. Aktaruzzaman Pramanik

We declare that this capstone project report entitled *BanglaFeel: Enhanced Bangla Back-Transliteration* is the result of our own work except as cited in the references. The capstone project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Masfiq Mahmud, 212014007

Dept. of CSE, University of Liberal Arts Bangladesh

Sumaiya Akter Iqra, 211014004

Dept. of CSE, University of Liberal Arts Bangladesh

Muhammad Anas, 212014020

Dept. of CSE, University of Liberal Arts Bangladesh

Rafit Ahmed Mullick, 212014038

Dept. of CSE, University of Liberal Arts Bangladesh

Anika Tasnim Bushra, 211014019

Dept. of CSE, University of Liberal Arts Bangladesh

Date: May 19, 2025



Department of Computer Science and Engineering
University of Liberal Arts Bangladesh
Mohammadpur, Dhaka - 1207

CERTIFICATE

This is to certify that the capstone project report entitled **BanglaFeel: Enhanced Bangla Back-Transliteration**, submitted by **Masfiq Mahmud** (Student ID: 212014007), **Sumaiya Akter Iqra** (Student ID: 211014004) and **Muhammad Anas** (Student ID: 212014020) **Rafit Ahmed Mullick** (Student ID: 212014038) **Anika Tasnim Bushra** (Student ID: 211014019) are undergraduate students of the **Department of Computer Science and Engineering** who have been examined. Upon recommendation by the examination committee, we hereby accord our approval of it as the presented work and submitted report fulfills the requirements for its acceptance in partial fulfillment for the degree of *Bachelor of Science in Computer Science and Engineering*.

Md. Aktaruzzaman Pramanik, Lecturer
Dept. of CSE, University of Liberal Arts Bangladesh

Ms. Rubaiya Hafiz, Senior Lecturer
Dept. of CSE, University of Liberal Arts Bangladesh

Shah Jafor Sadeek Quaderi, Lecturer
Dept. of CSE, University of Liberal Arts Bangladesh

Place: Dhaka

Date: May 19, 2025

ACKNOWLEDGEMENTS

We would like to express our deep and sincere gratitude to our research supervisor, *Md. Aktaruzzaman Pramanik*, for giving us the opportunity to conduct research and providing invaluable guidance throughout this work. His dynamism, vision, sincerity and motivation have deeply inspired us. He has taught us the methodology to carry out the work and to present the works as clearly as possible. It was a great privilege and honor to work and study under his guidance.

We are greatly indebted to our honorable teachers of the Department of Computer Science and Engineering at the University of Liberal Arts Bangladesh who taught us during the course of our study. Without any doubt, their teaching and guidance have completely transformed us to the persons that we are today.

We are extremely thankful to our parents for their unconditional love, endless prayers and caring, and immense sacrifices for educating and preparing us for our future. We would like to say thanks to our friends and relatives for their kind support and care.

Finally, we would like to thank all the people who have supported us to complete the project work directly or indirectly.

Masfiq Mahmud, Sumaiya Akter Iqra, Muhammad Anas, Rafit Ahmed Mullick and Anika Tasnim Bushra

University of Liberal Arts Bangladesh

Date: May 19, 2025

Dedicated to
my loving Mother and Dad
– *Masfiq Mahmud*

Dedicated to
my beloved family
– *Sumaiya Akter Iqra*

Dedicated to
my lovely family and
friends
– *Muhammad Anas*

Dedicated to
my Mother
– *Rafit Ahmed Mullick*

Dedicated to
my loving parents
– *Anika Tasnim Bushra*

ABSTRACT

Bangla, a widely spoken language, frequently appears online in its Romanized form, "Banglish," creating significant challenges for Natural Language Processing models, particularly in sentiment analysis, as they are typically trained on native-script Bangla or standard English. The objective of this study is to develop a proposed model, a back-transliteration system designed to accurately convert Banglish text into native Bangla script, thereby improving downstream NLP tasks for the project. The methodology employs a multiple encoder, single decoder architecture integrating character-level CNN and word-level LSTM encoders with a pre-trained T5 transformer decoder to capture both fine-grained spelling variations and broader contextual nuances. Trained and evaluated on the BanglaTLit dataset, the proposed model provides significant improvements over existing methods, achieving a BLEU score of 68.03%, ROUGE-1 F1 of 83.34%, ROUGE-2 F1 of 43.93%, ROUGE-L F1 of 83.38%, METEOR score of 81.55%, and a BERTScore F1 of 96.38% on the Test Data, effectively handling the inherent variability of Banglish. The proposed model successfully overcomes the challenges of transliteration by converting Romanized Bangla into native script.

Keywords: Banglish, Back-Transliteration, BiLSTM, CharCNN, T5, BanglaFeel, BanglaTLit Dataset

Contents

1	Introduction	1
1.1	Problem statement	2
1.2	Aims, objectives & Motivation	2
1.2.1	Aims & Objectives	2
1.2.2	Motivation	3
1.3	Project Specification	3
2	Literature Review	4
2.1	Overview	4
2.2	Terminology	15
2.3	Custom Encoder	16
2.4	T5 Decoder	17
2.5	Beam Search	17
3	Methodology	22
3.1	Method of the study	22
3.1.1	Dataset	23
3.1.2	Data Preprocessing	27
3.1.3	Data Splitting	27
3.1.4	Proposed Architecture	28
3.1.5	Model Training with Hyperparameter Tuning	29
3.1.6	Performance Evaluation	29
3.2	Proposed Custom Architecture Model	30
3.3	Project Budgeting	32

4	Result analysis	33
4.1	The <code>translate_text</code> Function	33
4.2	Performance Metrics	45
4.3	Transliteration Interface	48
5	Conclusions	49
5.1	Limitations	49
5.2	Future Work	50
5.3	Ethical Issues	50
5.4	Social Issues	51
5.5	Brief Summary	51
	References	53

List of Figures

1.1	Illustration of Bengali Non-Standard Transliteration for 'Let's Drink Juice.	2
3.1	Flowchart illustrating the methodology.	22
3.2	The BanglaTLit datasets.	25
3.3	Distribution of categories visualized as a donut chart.	26
3.4	Graphical illustration of proposed model architecture - BanglaFeel. .	31
4.1	Learning Curve-Loss and F1 Score.	37
4.2	Learning Curve - Loss and Precision.	38
4.3	Learning Curve - Loss and Recall.	39
4.4	Loss Curve.	40
4.5	Precision-Recall Curve.	41
4.6	Validation F1 Score.	42
4.7	Validation Precision.	43
4.8	Validation Recall.	44
4.9	BanglaFeel Transliteration Interface.	48

List of Tables

2.1	Comparative Study.	20
3.1	Transliterated Text and Corresponding Bangla Text	23
3.2	Transliterated Text and Corresponding Bangla Text	23
3.3	Transliterated Text and Corresponding Bangla Text	24
3.4	Project Budget Breakdown.	32
4.1	Performance Metrics of Test Data.	46
4.2	Comparison Performance Metrics of Test Data Between Models.	47

Chapter 1

Introduction

Bangla is one of the worlds most widely spoken languages, boasting over 250 million native speakers and a robust online presence where Banglish, the Romanized form of Bengali text, regularly appears. Existing Natural Language Processing (NLP) solutions, however, struggle to interpret Banglish effectively, particularly in sentiment analysis tasks, because most pre-trained models are trained on standard English or native-script Bangla and cannot readily accommodate the informal, inconsistent nature of transliterated text [Fahim et al. \(2024\)](#). Although recent research suggests that back-transliteration models can convert Romanized Bangla back into Bengali script for downstream applications, inadequate character-level modeling, and difficulty handling noisy, highly variable social media content. While datasets like BanglaTLit have emerged to support this endeavor, few existing approaches incorporate architecture designs that explicitly capture both character-level and word-level nuances in transliterated Bangla. Motivated by the need to reduce costly and time-consuming fine-tuning processes for sentiment analysis and other Bangla NLP tasks, this research introduces a translation model tailored to handle Banglish text. The model fuses a specialized custom encoder integrating a character-level CNN with a word-level LSTM with a pre-trained T5-based encoder-decoder. By capturing subtle spelling variations at the character level and contextual relationships at the word level, the custom encoder produces robust embeddings that facilitate more accurate Bangla script generation in the T5 decoder. We position this approach as a general-purpose transliteration layer that easily integrates with existing NLP pipelines, thereby boosting performance not only in sentiment analysis but also in a wide range of Bangla language processing tasks. Drawing on best practices in sequence-to-sequence modeling, leveraging large-scale resources such as BanglaTLit [Fahim et al. \(2024\)](#), and introducing dual encoder

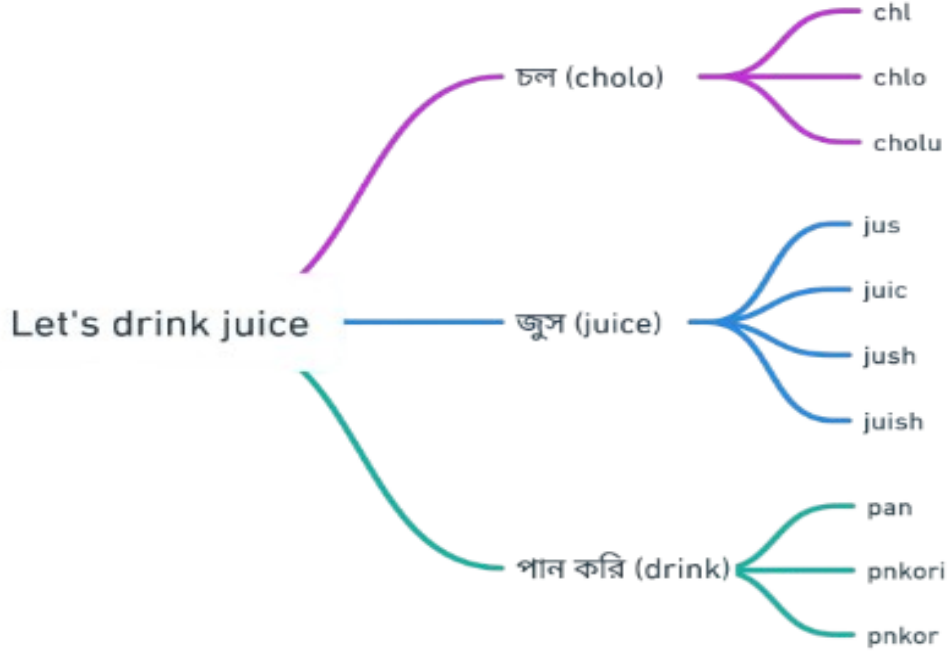


Figure 1.1: Illustration of Bengali Non-Standard Transliteration for 'Let's Drink Juice.

techniques for deeper linguistic representation, the framework offers empirical evidence that accurate back-transliteration is crucial for improving downstream NLP performance across the Bangla language landscape.

1.1 Problem statement

The prevalence of Romanized Bangla ('Banglish') in digital communication significantly challenges existing NLP systems, which are typically trained on formal Bangla or English. Current NLP techniques struggle to process it effectively without resource-intensive fine-tuning, leading to reduced accuracy.

1.2 Aims, objectives & Motivation

1.2.1 Aims & Objectives

The aim of this project is to develop a robust and accurate system for back-translitterating Romanized Bangla text into native Bangla script, leveraging advanced natu-

ral language processing techniques to handle the challenges of phonetic variations, informal spellings, and inconsistent use of diacritics. To design and implement a custom Encoder that captures both character-level and word-level features of Romanized Bangla text. To make the system publicly accessible by releasing the code and research paper for future research and development.

1.2.2 Motivation

Inspiration comes from the widespread use of Romanized Bengali (Banglish) in social communication and the lack of strong tools for processing informal text of this sort. Social media platforms are making extensive use of Banglish these days, even though its non-standard and phonetic form renders it highly challenging for NLP tools like content moderation and sentiment analysis [Fahim et al. \(2024\)](#). Current models are not able to back-transliterate Banglish into native script properly and limit their ability to analyze and understand user-posted text [Haider et al. \(2024\)](#). With the design of a multiple encoder single decoder model which includes character level as well as word level features, this project aims to enhance back-transliteration accuracy for Banglish. The use of BanglaTLit corpus resolves resource scarcity for low-resource languages including Bangla [Fahim et al. \(2024\)](#). Not only does it enhance NLP tools for Bangla but it also allows businesses, researchers and organisations to make use of these tools in a more efficient way to engage Bengali-speaking cyberspaces in initiating a culture of inclusiveness and enhanced e-participation in today's ever-electronic age.

1.3 Project Specification

The project focuses on to design a multi-encoder single decoder model for back-transliterating Romanized Bangla or Banglish into native script Bangla. The system deploys a custom Encoder for word and character level processing and a T5 Encoder pre-trained for context embeddings. The designed architecture is meant to create accurate and natural transliteration and capture structural and semantic richness in input text while achieving balanced output length and context integrity. The architecture is meant to outperform baselines in its capacity for robust handling of diverse transliteration patterns and its scalability for large datasets in an efficient manner. The architecture is generalizable in nature, can be trained within reasonable numbers of epochs, and can be scaled for different batch sizes for best performance.

Chapter 2

Literature Review

2.1 Overview

Our aim in research is to create back-transliteration of Romanized Bangla or Banglish into native-script Bangla for enhanced use in natural language processing (NLP) operations, including sentiment analysis. To enable this overview, we employed a systematic approach in order to have a complete and objective overview of available extant knowledge. We searched for peer-reviewed articles and conference papers published within the last five years, prioritizing studies that address Bangla or similar low-resource languages, propose innovative solutions to transliterated text processing, and contribute to advancing NLP methodologies. Our process involved querying academic databases such as Google Scholar, IEEE Xplore, and arXiv with keywords including "Bangla transliteration," "Banglish sentiment analysis," "NLP for low-resource languages," and "encoder-decoder architectures." We also considered availableness of datasets and opportunity for reproducibility as key determinants in the selection of applicable literature. Our survey encompasses a diverse range of topics relating to our research objectives, including transliteration methods, sentiment analysis for low-resource and code-mixed settings, and cutting-edge NLP models including encoder-decoder models and transformer models. Reading through these works, we outline shortcomings of prior methods such as being unable to manage informal spelling, phonetic spelling variations, and code-mix in Banglish and propose gaps in studies which "BanglaFeel" is trying to bridge. The insights gained from this review directly inform the design of our proposed multiple encoder, single decoder architecture, which integrates character-level Convolutional Neural Networks (CNN) and word-level Long Short-Term Memory (LSTM) networks with a pre-trained T5 transformer to improve

back-transliteration accuracy. It was aimed to establish a solid foundation for our project by contextualizing our contributions within the broader landscape of Bangla NLP research and demonstrating how our proposed model advances the Banglish Back-Transliteration efficiently.

[Fahim et al. \(2024\)](#) introduces "BanglaTLit," a significant new benchmark dataset designed for the back-transliteration of Romanized Bangla, addressing a critical need in NLP for low-resource languages. The authors highlight the inadequacy of existing datasets for this task, which aligns with your problem statement that current sentiment analysis models struggle with Banglish transliterated text. BanglaTLit, comprising 42.7k manually annotated and expert-verified samples, alongside a pre-training corpus (BanglaTLit-PT) of 245.7k samples, offers a robust resource for training and evaluating models. Furthermore, the paper demonstrates that further pretraining transformer encoders on BanglaTLit-PT achieves state-of-the-art performance in several romanized Bangla classification tasks, highlighting the potential for improved accuracy in related tasks like sentiment analysis. By providing a large-scale, diverse, and high-quality dataset specifically for Romanized Bangla, this research directly supports the need for specialized models capable of handling the unique challenges of Banglish, as identified in your problem statement, and offers a pathway to developing more effective sentiment analysis tools for Bengali-speaking online users.

[Li et al. \(2020\)](#) investigates the effectiveness of multi-encoder approaches in document-level neural machine translation (NMT), particularly focusing on how the context encoder impacts translation quality. The study reveals that multi-encoder models, even when fed with random or fixed context, show improvements similar to those using actual context, suggesting that the context encoder acts more like a noise generator for robust training rather than a true context interpreter. This aligns with current models are inadequate for accurately interpreting complex language structures like transliterated Banglish, as they might struggle to capture the true contextual nuances necessary for accurate sentiment analysis. The paper's emphasis on the limitations of current models in genuinely understanding context, even in a related field like NMT, further supports the need for improved sentiment analysis models specifically designed to handle the unique characteristics of Banglish transliterated text.

Baruah et al. (2023) address the challenge of effectively transliterating Assamese text from the noisy, non-canonical forms commonly found on social media platforms. They note that existing transformer-based transliteration models, typically trained on clean, manually transliterated (canonical) datasets, fail to accurately process the highly ambiguous and noisy characteristics of user-generated content. This shortcoming hinders various downstream natural language processing tasks that depend on accurate transliteration. To tackle this issue, the authors conduct a systematic evaluation of fifteen different experimental setups, combining both canonical and non-canonical datasets. Their findings reveal that models trained exclusively on either clean or noisy data perform inadequately when exposed to the other type. Yet a model based on both datasets performs even better, which suggests that the fusion of noisy and labeled data is responsible for building robust transliteration models able to resist problems because of Assamese social network text.

Hasan et al. (2023) address the task of IPA transcription generation for text in Bangla, challenging due to the extensive phonetic mappings of the language that are hard to represent through manual modeling. Resources for text-to-IPA conversion from Bangla are limited, being harmful for text-to-speech systems and speech therapy and language research. To overcome this task, the authors use a Transformer-based sequence-to-sequence model on a character level for IPA transcription generation for words in Bangla leveraging low word-context effect in IPA representation. Manual handling for out-of-dictionary words and for punctuations is also employed in reducing computational complexity based on an assumption that these would not get effectively learned by the model as distinct from words in Bangla. With it, they are able to get a light model of 8.5 million parameters with a word error rate of 0.10582 and ranking first in DataVerse ChallengeITVerse 2023. The authors state that word-level IPA generation and an IPA dictionary can ensure optimal performance and resource utilization. Their Transformer-based solution thus offers an efficient solution for text-to-IPA transcription in Bangla for developing tools and applications for Bangla.

Mondal et al. (2024) address the problem of English-Bengali machine translation (MT) under the scenario of low-resource languages where large parallel corpora are not available. Current MT models for language pairs such as English-German are

complemented by rich corpora of data, but English-Bengali MT is challenging because of data scarcity, which influences the performance of translation models and has a cascading effect on downstream language processing applications dependent on accurate translations. For improving translation quality, the researchers resort to back-translation by creating pseudo-parallel corpora from Bengali monolingual data back-translated into English. By adding the pseudo data into the original data set, they are essentially enlarging the training corpus. The authors experiment with Long Short-Term Memory (LSTM) models and Transformer models, and they observe that the Transformer scores much higher than LSTM in this setting. In addition, they utilize decoding techniques such as top- k random selection and random selection using temperature T during back-translating in order to render the resulting pseudo data more trustworthy. Their experimentation is found to considerably boost BLEU scores and thus overall translation effectiveness through back-translation, and even further through random selection based on an optimal temperature $T = 0.5$. The study demonstrates that exploiting back-translation with optimal decoding techniques is an effective way for enhancing English-Bengali MT that is much needed for building trustworthy language processing tools for low-resource languages.

[Nath et al. \(2024\)](#) tackle technical term and proper name translation in English-Assamese MT since it is a low-resource pair. NMT models cannot handle words in this manner since they must be transliterated for equivalent sounds rather than being translated since it would undermine general translation quality. To solve this issue, the researchers propose a hybrid encoder-decoder attention-based transliteration system for English-Assamese and integrate it into their NMT system. Through systematic evaluation using metrics like BLEU, METEOR, and TER, they find that models trained exclusively on either translation or transliteration tasks perform inadequately when dealing with the other type. However, the integrated model that combines both translation and transliteration capabilities demonstrates improved performance, suggesting that such integration is essential for building robust NMT systems capable of handling the complexities of Assamese translation.

[Talafha et al. \(2021\)](#) research on Arabic transliteration has explored various approaches, including rule-based systems, statistical machine translation (SMT), and

more recently, neural networks. While some work exists on dialectal Arabic transliteration, resources are limited, particularly for Jordanian and other spoken dialects. Previous studies have employed techniques like conditional random fields and recurrent neural networks (RNNs), including LSTMs, but there’s a need for further investigation into improving accuracy and handling the nuances of spoken Arabic, which often deviates significantly from Modern Standard Arabic in terms of vocabulary, morphology, and pronunciation. This paper aims to address this gap by proposing an attention-based LSTM model for Arabic-to-Arabizi transliteration.

Haider et al. (2024) introduces BANTHI, a new multi-label hate speech dataset for Bangla, addressing the scarcity of labeled data for this low-resource language. The dataset comprises 37,350 samples annotated across seven hate categories and includes both native Bangla script and its transliterated Romanized form (Banglish). The authors evaluate several pre-trained Language Models (LMs), including monolingual, multilingual, and character-based models, using fine-tuning, few-shot, and prompting strategies. They also explore a translation-based approach, converting Bangla text to English before classification, and demonstrate the effectiveness of prompting-based methods, particularly with GPT-3, achieving state-of-the-art results on BANTHI. Furthermore, they incorporate explainable AI (XAI) techniques to provide insights into model predictions and address the need for transparency in hate speech detection.

Islam et al. (2022) emphasizes the scarcity of robust lemmatization resources for Bangla. Existing methods often rely on rule-based approaches or limited datasets, resulting in lower accuracy. This scarcity directly impacts downstream tasks like sentiment analysis. If the fundamental task of lemmatization is inaccurate, the subsequent analysis of sentiment will likely be flawed. The paper also points out that existing lemmatizers for morphologically rich languages like Bangla often struggle with complex word forms and variations, further hindering accurate sentiment analysis. Therefore, the development of a more accurate lemmatizer, as proposed in the paper, is a necessary step towards improving the performance of sentiment analysis models for Bangla, and indirectly, for Banglish text, as accurate lemmatization of the original Bangla script could improve the training data for Banglish sentiment models.

Shiam et al. (2023) offers relevant insights within the broader context of processing Bengali-language data online. Our research problem focuses on the inadequacy of current sentiment analysis models to accurately interpret Banglish, the Romanized form of Bengali used extensively online. This paper highlights the challenges inherent in bridging the gap between English and Bangla, a crucial stepping stone towards tackling the complexities of Banglish. Specifically, the paper’s exploration of a neural machine translation (NMT) model for English-Bangla translation underscores the difficulties in capturing nuanced meanings and diverse vocabulary between these two languages. This directly relates to our problem statement, as the inaccuracies in translating between English and Bangla can be exacerbated when dealing with the informal and often inconsistent nature of Banglish. The paper’s use of a large dataset (CCMatrix) for training the NMT model also emphasizes the need for substantial, Bangla-focused datasets for training effective sentiment analysis models, a resource currently lacking for Banglish text. In essence, the paper indirectly supports our problem statement by demonstrating the existing complexities in automated processing of Bengali-derived text, even in a more structured form (Bangla vs. Banglish), highlighting the need for further research into understanding and processing the nuances of Banglish for accurate sentiment analysis.

Raihan et al. (2023) address a significant gap in natural language processing (NLP) concerning offensive language detection in transliterated and code-mixed text, specifically focusing on Bangla. Transliterated and code-mixed languages pose unique challenges for NLP systems due to their non-standardized spelling and the blending of multiple languages within a single text, which is common in multilingual societies and prevalent on social media platforms. They introduce TB-OLID, a novel dataset comprising 5,000 manually annotated Facebook comments written in transliterated Bangla and code-mixed Bangla-English. This is the first of its kind corpus for Bangla and will enable research in detecting offenses in code-mixed and transliteration setups.

Vania (2020) examines the extent to which morphological information is encoded in typologically rich languages by character-level neural network models. Their article examines in a critical manner if morphological information is learned implicitly by models and if including explicit morphological information in NLP systems is still necessary. They point out that while character models are an extremely useful resource

in NLP, particularly in rich morphological languages, they do not make morphological information unnecessary. The thesis promotes a balanced solution that can take advantage of data-driven learning and linguistic knowledge in an attempt to remedy the challenge of human language in computational models. The study presents a contribution towards general subword representation consideration in NLP and paves avenues towards future morphologically conscious processing of language.

[Salaev et al. \(2022\)](#) have given an automatic transliteration system that is capable of transliterating words in Uzbek script from one script to another. The rule-based approach involves character mapping and replacement processes for performing transliteration and tackling the issue of multiple scripts in Uzbek. The solution applied in this system can possibly be highly applicable in projects involving back-transliteration of Banglish into native script since both involve switching from one script to another while wholly avoiding loss of linguistic accuracy.

[Tareq et al. \(2023\)](#) respond to sentiment analysis of code-mixed text in Bangla-English by proposing data augmentation methods that promote cross-lingual context understanding. Given that conventional NLP models deteriorate in a context of informal spelling and phonetic spelling common in social media text in Banglish, the authors respond by proposing methods for cross-lingual word replacement for data augmentation purposes in order to make the model more robust. By substituting words in code-mixed sentences for semantic translations in either of the languages, the authors are able to enhance variability in training data and allow for improved code-mixed representation learning in the model. The experimental results indicate that the data augmentation approach of the study significantly improves the performance of sentiment models for sentiment classifications for code-mixed datasets in Bangla-English, proving its effectiveness in overcoming code-mixed and transliteration text intricacies.

[Alam et al. \(2024\)](#) have released a paper "BSENTMIX: Dataset for Sentiment Analysis of Bangla-English Code-Mixed Texts," in which they have introduced a large code-mixed sentiment analysis corpus trying to address the code-mixed text processing issue in Bengali and English. The corpus includes over 20,000 labeled data for four sentiment categories (negative, positive, neutral, and mixed sentiments). The researchers use mBERT and BERT transformer models for classification purposes

and achieve remarkable performance improvement through code-mixed fine-tuning. The researchers propose using data augmentation methods for making models robust, particularly for underclass sentiment tags. The study is particularly interesting for back-transliteration system building due to its solution of preprocessing, linguistic variation, and code-mixed dataset processing complexity, which is of similar complexity for processing Banglish transliteration.

[Rahman et al. \(2024\)](#) compared sentiments in eLearning websites in the presence of Bangla and Romanized Bangla text and transformer models in 2024. With an ever-growing requirement for ongoing skill acquisition in association with rapid technological advancement, eLearning is accepted by the authors as an innovative pedagogy that provides lessons through electronic means. Their article highlights the performance of transformer models in processing both native and Romanized Bangla text for performing sentiment analysis even though natural language processing makes it challenging for low-resource languages. Comparing model performances for native and Romanized scripts, the research presents major findings that are directly applicable for projects targeting Romanized Bangla processing and prove transformer architecture efficient for processing such data.

[Duong & Nguyen-Thi \(2021\)](#) offer an extensive overview of preprocessing methods and data augmentation techniques applicable for sentiment analysis. Their discussion aptly tackles the task of coping with informal and irregular text data by highlighting the role played by data cleaning and normalization towards improving analytical accuracy. Duong and Nguyen-Thi discuss a range of preprocessing techniques that are currently in use, including negation and intensification treatment necessary for sentiment interpretation in informal text. Besides these processes, they highlight data augmentation techniques that enable new data generation from available datasets and thus increase training set size without requiring contributions from users and thus overcoming restricted labeled data limitations in supervised scenarios. The strategies are highly applicable for those Banglish projects involving Bengali-English interaction that make preprocessing challenging. The use of strategies described by Duong et al. can significantly advance preprocessing for sentiment analysis in similar multilingual and informal contexts.

[Sultana & Mamun \(2024\)](#) have conducted a study on "Data-Augmentation for Bangla-English Code-Mixed Sentiment Analysis: Enhancing Cross Linguistic Contextual Understanding" where they address the problem of informal spellings and spelling variation in Banglish. Employing cross-lingual word replacement techniques and data augmentation, they seek to enhance the cross-linguistic contextual awareness of sentiment analysis models in applying them on code-mixed data in Bangla-English. With the inclusion of synthetic data through augmentation techniques, they enhance the ability of the model in comprehending and labeling sentiments from code-mixed data. The methods in their study have direct applicability in building back-transliteration systems as they establish viable mechanisms of addressing issues of informal spellings and spelling variation in capturing sentiments in mixed-language datasets.

[Basri et al. \(2021\)](#) performed sentiment analysis of Bang-lish text, i.e., Bangla language in Roman script, using deep learning methods. They experimented with different neural network models with attention mechanisms to attempt to capture sentiments in informal and code-mixed Bang-lish revelations. Their work proves that deep learning models can effectively tackle the complexity of Romanized Bangla text for obtaining meaningful outputs, serving as a blueprint for developing encoder-decoder models in sentiment analysis research targeting Bang-lish or other low-resource Romanized languages.

[AlHazzani et al. \(2022\)](#) provided the Sa'7r dataset of 19,810 Saudi dialect tweets annotated for detection of irony, of which 8,089 are ironic. The informal and dialectal Arabic social text processing issue was tackled in their research by applying best-of-breed preprocess strategies in handling slang words, colloquials, and linguistic idiosyncrasies characteristic of the Saudi dialect. Their approach involved data cleaning, text normalization, and processing abnormal features of the dialectical type that can make for very useful hints in constructing successful preprocess pipelines for projects involving informal or dialectical textual data.

[Iqbal et al. \(2022\)](#) introduce BEmoC, a corpus designed for emotion detection in Bengali texts. When they found out that resources for labeling of emotions in low-resource languages like Bengali were scarce, the authors developed a large corpus involving a large number of emotional words in formal and informal contexts. The

corpus is based on annotated text for model development and evaluation for natural language processing tasks for emotion detection. With this effort, the study resolves major issues in processing difficult emotional content and is highly beneficial for systems like back-transliteration systems that must handle emotional and informal text in Bengali.

Patwa et al. (2020) conducted a study, "Comparative Analysis of Pre-Trained Transformer Models for Sentiment Analysis," in which different transformer models including mBERT, RoBERTa, and XLM-R are compared based on performance in low-resource and multilingual settings. The experiment checks performance on diverse datasets of different languages and scripts on transliterated and code-mixed text. Through fine-tuning tests and studies on adaptability in models for domain-specific purposes, researchers indicate interesting aspects of transformer models for sentiment analysis in terms of advantages and disadvantages. The research establishes transformer models as superior compared to conventional means of machine learning in handling rich linguistic phenomena, code-switching, and transliteration. The research comes in useful while building back-transliteration systems for Romanized Bangla (Banglish) because in the study, transformer models' ability in handling context features and processing multilingual informal datasets is recorded.

Hoque & Salma (2023) propose a study called "Detecting Depression Levels from Bengali Social Media Text," through which they address the difficulties related to sentiment and emotional analysis in low-resource languages like Bengali. The researchers built a dataset split into four depression levels:- non-depressive, mild, moderate, and severe, and tested through several machine learning and transformer-based models, including XLM-RoBERTa, for sentiment identification from social text. Preprocessing operations like tokenization and stemming were applied to tackle social text's informal nature. The experiment demonstrates the versatility of current transformer models in processing emotional content in under-resourced languages in a rich and subtle manner, a function that can greatly aid back-transliteration systems for informal and emotional processing of Banglish text.

Das et al. (2021) carried out a study captioned " Bangla Abusive Text Detection Using Machine Learning ", in which they explain the task of detecting abusive text

in transliterated and informal Bangla. During the course of the study, researchers amassed an extensive corpus of both abusive and non-abusive social media posts that were all hand annotated carefully. The researchers compared and carried out experimentations based on different machine models, including SVM and random forest models. The major preprocessing operations carried out included stemming, tokenization, and removal of stop words, all of which helped in improving the ability of models in extracting data from transliterated and noisy text. The findings from this study in terms of preprocessing methods and corpus construction for informal texts in Bangla have far reaching implications for improving back-transliteration systems for tackling similar problems in Banglish. Critical preprocessing techniques applied included stemming, tokenization, and removal of stop words that all improved models' ability in handling noisy and transliterated text. The findings of this study in regards to preprocessing methods and building datasets for informal Bangla text are useful for improving back-transliteration systems for tackling similar issues in Banglish.

[Shibli et al. \(2023\)](#) published a research study titled "Automatic Back Transliteration of Romanized Bengali (Banglish) into Bengali," which deals with back-transliteration from Romanized Bengali text into Bengali script. The researchers applied a seven-stage pipeline that incorporates nine open-source transliteration tools and tested their approach on a corpus of 1,000 Romanized Bengali texts. The pipeline includes a sequence of processing steps that are particularly aimed at informal and transliterated aspects of Romanized Bengali text and have achieved performance rating of BLEU-1 (81.28) and Word Error Rate (29.21). The novelty in this study involving developing a systematic pipeline and its evaluation on a domain-specific corpus provides a baseline framework for automating back-transliteration and thus provides a solution for informal and transliterated Bangla text problems in electronic communications.

[Nanayakkara et al. \(2022\)](#) have released a "Context Aware Back-Transliteration from English to Sinhala," a study that deals with back-transliteration of Sinhala text in English script, a common issue in posts in social networking sites. The researchers formulated a back-transliteration system around a Transliteration Unit (TU) that relies on context for improving transliteration of Romanized-Sinhala text back into its native Sinhala scripts. Segmenting text in parts called Transliteration Units and

analyzing context under which words are being used makes it easier to minimize ambiguity due to more than one Sinhala word having similar Romanized representations. The context-aware method is a major innovation compared to available Sinhala Natural Language Processing tools and thus enhances their ability to process large sets of texts in electronic form that have Romanized-Sinhala content. Applying Transliteration Units and context analysis in this study is an innovative approach for developing equivalent systems for other languages that also have similar problems in text transliteration.

2.2 Terminology

The Bengali phrase "ফাইভার এ দুকেছি ৩ মাস হলো..." (Fiverr e dhukesi 3 mash holo) which translates to I've been on Fiverr for 3 months now in English) was used as a representative input to demonstrate the effectiveness of various tokenization and encoding techniques tailored for the Bengali language.

Data Preprocessing: The input text "Fiverr e dhukesi 3 mash holo" was first prepared for processing by applying three distinct tokenization methods:

- **T5 Tokenizer:** We employed the Text-to-Text Transfer Transformer Tokenizer to subword tokenize the input text in a form that can be processed by the T5 model. The sentence is split into subword pieces by this tokenizer in order for different word forms and out-of-vocabulary words to be processed by the model [Roberts & Raffel \(2020\)](#).
- **BanglaBERT Tokenizer:** To leverage features of a particular language, we employed the use of BanglaBERT Tokenizer, designed for Bengali text. BanglaBERT is a Bengali corpus-pretrained variant of the BERT model that enables rich representations for Bengali words and word sequences [Mahmud et al. \(2024\)](#).
- **CharCNN Tokenizer:** Besides that, we have also employed a Character-level Convolutional Neural Network (CharCNN) Tokenizer for tokenizing text into individual characters. The approach preserves more fine-grained lexicon information best for capturing morphological differences in Bengali [Ye et al. \(2024\)](#).

Encoding Techniques: After tokenization, the tokens are transformed into vectors through different encoding methods:

- **T5 Encoder:** T5 Tokenizer tokenized input was given as an input to T5 Encoder for token mapping into fixed-size representations of semantic meaning of text [Roberts & Raffel \(2020\)](#).
- **WordLSTM Encoder:** Tokens of the BanglaBERT Tokenizer were fed into a Word-level Long Short-Term Memory (WordLSTM) Encoder. The encoder incorporates sequential word relation, encoding word context and word relation [Hori et al. \(2018\)](#).
- **CharCNN Encoder:** A CharCNN Encoder has been employed for character tokenization. The encoder carries out convolution operations on the character sequence for extracting local pattern and n-gram features in order to emphasize morphological and subword representation [Ye et al. \(2024\)](#).

2.3 Custom Encoder

Custom Encoder (WordLSTM, CharCNN) is our custom-designed architecture for handling Romanized Bangla text orthographic complexity. The model processes on both word and character levels for retrieving word and character level fine-grained patterns and context clues. On a word level, input text is divided into individual characters and each one is given an individual embedding vector. The embeddings are processed through a Convolutional Neural Network having multiple kernel sizes (e.g., 2, 3, and 4) for feature extraction in an n-gram. The CNN output for a character sequence $c = \{c_1, c_2, \dots, c_k\}$ can be expressed as:

$$\text{CNN}(c) = \text{MaxPool}(\text{ReLU}(\text{Conv}(c)))$$

where Conv represents the convolutional operation, ReLU is the activation function, and MaxPool is the max-pooling operation. At the word level, the input text is tokenized into words, and each word is mapped to a dense embedding vector. These embeddings are processed by a Bidirectional LSTM to capture contextual relationships between words. The LSTM output for a word sequence $w = \{w_1, w_2, \dots, w_n\}$ can be expressed as:

$$\text{LSTM}(w) = \text{BiLSTM}(\text{Embed}(w))$$

where Embed represents the word embedding layer, and BiLSTM denotes the bidirectional LSTM operation. The final output of the Custom Encoder (WordLSTM, CharCNN) is a combination of character-level and word-level representations, which are concatenated to produce a unified feature vector for each token. This dual-level approach ensures that the Custom Encoder (WordLSTM, CharCNN) effectively captures both local phonetic patterns and global contextual information, making it robust for handling the variability in Romanized Bangla text.

2.4 T5 Decoder

Once the aggregate representations are ready, a T5 decoder attends over the combined encoder states to generate Bangla tokens in an autoregressive manner. During training, we apply standard crossentropy between the predicted tokens and the reference Bangla text. Formally, if y is the correct token at time t , the decoders negative loglikelihood loss can be expressed as:

$$L(\theta) = - \sum_t \log P_\theta(y_t \mid x, y_0, \dots, y_{t-1}) \quad (2.1)$$

where x is the input transliterated sentence, $y_0 \dots y_{t-1}$ are the previously decoded tokens, and θ denotes all trainable parameters (encoders, decoder, embeddings). After that, the output will go through a "de-tokenizer" that merges T5 subwords into full Bangla words ("ফাইভার", "এ", "টুকেছি", "ও", "মাস", ...).

2.5 Beam Search

To enhance our model's ability to produce coherent and contextually relevant text we implemented Beam Search algorithm integral to the decoding phase of our text generation model, designed to identify the most probable sequence of tokens for tasks such as text generation, translation, or summarization. It systematically explores multiple candidate sequences and selecting the one with the highest cumulative probability. In the architecture, Beam Search follows the T5 Decoder, which generates token probabilities based on aggregated features derived from the T5 Encoder, WordLSTM

Encoder, and CharCNN Encoder. These encoders process the input text at token, word, and character levels, respectively, creating a comprehensive representation that informs the decoding process. Beam Search efficiently navigates the vast search space of possible sequences, ensuring high-quality output by maintaining a fixed number of promising candidates, or "beams," at each step [Kasai et al. \(2022\)](#).

Beam Search for Sequence Generation

In our T5-based encoder-decoder framework, the decoder employs Beam Search to generate accurate Bangla sequences from Romanized Bangla inputs, leveraging aggregated representations from the T5 Encoder and a Custom Encoder (WordLSTM, CharCNN). Beam Search, a heuristic search algorithm, balances exploration and efficiency by maintaining k candidate sequences (beam width) at each decoding step, exploring multiple hypotheses in parallel. This mitigates the limitations of greedy search, enhancing back-transliteration accuracy.

As depicted in Figure 5.1., for a Romanized Bangla input x (e.g., corresponding to "ফাইভার", "এ", "টুকেছি", "ও", "মাস",), the encoder generates aggregated representations $z = \{z_1, z_2, \dots, z_n\}$, integrating token-level features from the T5 Encoder with character- and word-level features from the Custom Encoder. The T5 decoder attends over z to compute the conditional probability:

$$P(y_t \mid x, y_1, \dots, y_{t-1})$$

for each token y_t at step t . Beam Search retains k sequences, scoring them via cumulative log probabilities:

$$\log P(S \mid x) = \sum_{i=1}^t \log P(y_i \mid x, y_1, \dots, y_{i-1})$$

where $S = (y_1, y_2, \dots, y_t)$. The process iterates until an end-of-sequence token s is generated, selecting the highest-scoring sequence.

The process outline is as follows:

1. **Initialization:** Begin with $[s]$.
2. **Expansion:** For each sequence S_j in the beam, compute $P(y_t \mid x, S_j)$ for all next tokens.
3. **Scoring:** Update scores:

$$\text{Score}(S_j + y_t) = \text{Score}(S_j) + \log P(y_t \mid x, S_j)$$

4. **Selection:** Retain the top k sequences.
5. **Termination:** Select the highest-scoring completed sequence upon generating s .

With $k = 5$, Beam Search explores diverse hypotheses for $Y = (\text{"ফাইভার"}, \text{"এ"}, \text{"টুকেছি"}, \text{"ও"}, \text{"মাস"},)$, ensuring contextual alignment.

Selection of the best outcome, which is the optimal sequence Y^* , maximizes:

$$Y^* = \arg \max_Y \sum_{t=1}^T \log P(y_t \mid x, y_1, \dots, y_{t-1})$$

After de-tokenization, this yields the intended Bangla text. A beam width of $k = 5$ optimizes accuracy and efficiency, as validated experimentally.

Current models have difficulties in handling transliterated text where the content differs in spelling, grammar and uses code-mixing. Our research aims to bridge the identified research gaps by developing a multiple encoder, single decoder architecture specifically for Banglish back-transliteration, leveraging pre-trained transformer models and custom encoders to capture both character and word-level information. The impact of this improved back-transliteration on Bangla sentiment analysis will be rigorously evaluated. Building on previous work conducted in the NLP course, this

project will create a specific model which would use byte-pair encoding and attention mechanisms in order to overcome the issue of limited lemmatization resources and the fact that Banglish is quite informal to increase sentiment analysis accuracy.

Features	Fahim et al. (2024)	Li et al. (2020)	Baruah et al. (2023)	Hasan et al. (2023)	Mondal et al. (2024)	Nath et al. (2024)	Talafha et al. (2021)	Haider et al. (2024)	Islam et al. (2022)	Shiam et al. (2023)	Raihan et al. (2023)	Vania (2020)	Salaev et al. (2022)	Tareq et al. (2023)	Alam et al. (2024)	Rahman et al. (2024)	Duong & Nguyen-Thi (2021)	Sultana & Mamun (2024)	Basri et al. (2021)	Proposed Model
Transliterated Text	✓	✗	✓	✗	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Multiple Encoder	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
Back-Transliteration	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓
Informal/Code-Mixed Bangla Handling	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓	✓	✗	✓	✓	✓
CharCNN+BiLSTM+T5 Transformer	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Table 2.1: Comparative Study.

This table presents a comparative overview of various research efforts within a specific area, likely related to processing different forms of the Bangla language. It serves as a visual summary to quickly understand how different approaches, developed by researchers like Fahim et al., Li et al., and others over the years from 2020 to 2024, stack up against each other based on a set of key capabilities or technical characteristics. The table essentially maps out the landscape of recent work. Along the vertical axis, it lists important functionalities or design choices that are relevant in this research domain. These include the ability to work with transliterated text (like Bangla written using English letters), the capacity for back-transliteration (con-

verting that text back to the original script), and the robustness to handle informal or code-mixed Bangla (casual language or sentences mixing Bangla and English). It also considers architectural aspects, such as whether a method uses multiple encoders for processing input or employs a specific advanced combination of techniques like CharCNN, BiLSTM, and T5 Transformers. Each column represents a specific published study or method, including one labelled "Our (Proposed)," which is the new method being introduced by the authors of the document containing this table. By reading across a row, you can see which studies implemented a particular feature (indicated by a checkmark ✓) and which did not (indicated by a cross ✗). Conversely, reading down a column gives you a profile of a single method, showing the collection of features it possesses. The primary purpose of presenting the information this way is to highlight the contributions and potential advantages of the "Proposed Model" method. The table visually argues that while previous methods incorporated some of these desirable features, the proposed approach is more comprehensive, integrating all the listed capabilities handling various text forms, dealing with language mixing, and potentially using a sophisticated underlying architecture which distinguishes it from the prior work shown. It's a way to quickly demonstrate the novelty or completeness of the new research compared to existing solutions in the field.

Chapter 3

Methodology

The methodology is designed to address the challenges of back-transliterating Romanized Bangla (Banglish) into native-script Bangla. It encompasses two primary components: i) Developing a Custom Encoder for Transliterated Bangla and ii) Integrating the Custom Encoder with a Pre-trained T5-based Encoder-Decoder Framework. This dual approach leverages both character-level and word-level representations to enhance the accuracy and robustness of the transliteration process.

3.1 Method of the study

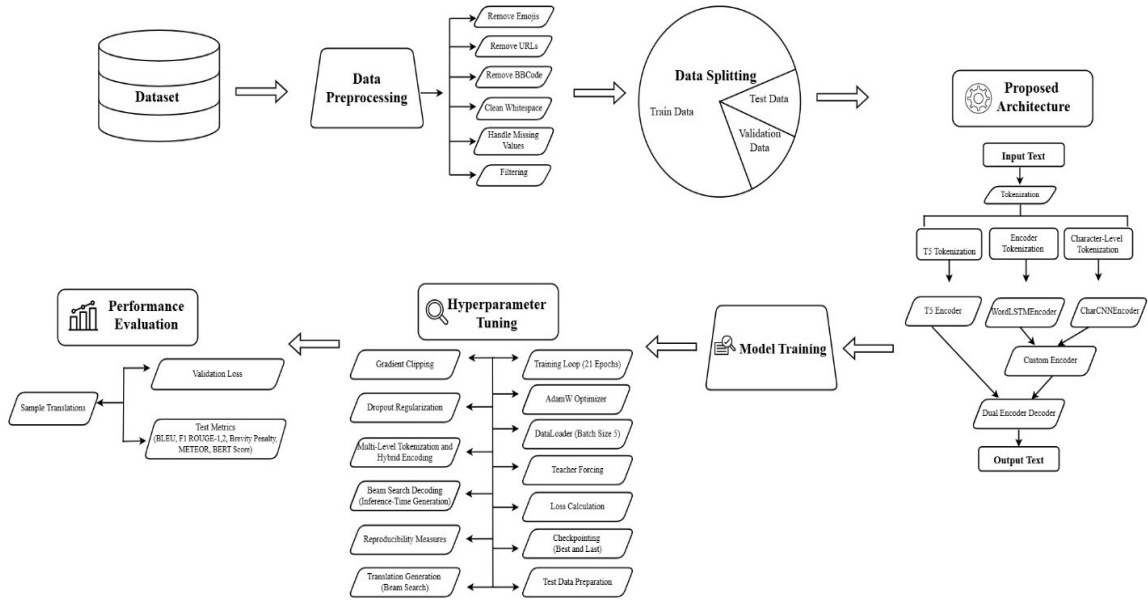


Figure 3.1: Flowchart illustrating the methodology.

3.1.1 Dataset

This research draws on the BanglaTLit framework (Fahim et al. (2024)), a resource specifically designed to handle Romanized Bangla text (often referred to as Banglish). Two key components form the backbone of this dataset: BanglaTLit-PT, which serves as a large-scale pre-training corpus of Romanized Bangla, and BanglaTLit, a set of back-transliteration pairs containing both the transliterated (Romanized) version of Bangla text and its corresponding native-script form. These resources provide sufficient diversity and scale to develop, train, and evaluate models aiming to bridge Banglish and native Bangla text processing. The dataset contains the following features: **id**, **text_transliterated**, **text_bengali**.

1. BanglaTLiT_validation_data

id	text_transliterated	text_bengali
d80b6cd1-d0d7-4d79-bc80-aa8566765110	Karo kaj korle bolben please	কারো কাজ করলে বলবেন প্লিজ
87524d44-1084-4439-b018-b8224859eac6	GameLoop use	গেম লুপ ইউজ করেন।

Table 3.1: Transliterated Text and Corresponding Bangla Text

2. BanglaTLiT_train_data

id	text_transliterated	text_bengali
f5afeacc-e452-497d-80d4-daf5c3238368	oi mia watermark a click korle subscribe hobe na?? title edit koren vaotabaji kothakar	ঐ মিয়া ওয়াটারমার্ক এ ক্লিক করলে সাবসক্রাইব হবে না? টাইটেল এডিট করেন ভাওতাবাজি কোথাকার
14a899f1-e09d-4fc2-ad2d-f92075d5992a	Eta niyeo post korte hoy??	এটা নিয়েও পোস্ট করতে হয়??

Table 3.2: Transliterated Text and Corresponding Bangla Text

3. BanglaTLit_test_data

id	text_transliterated	text_bengali
78c56009-fc89-4948-a1f5-c80ed6d6cf7a	Tnx bro ato sundor akta gift dewar jonno	থ্যাংক্স ব্রো এত সুন্দর একটা গিফট দেওয়ার জন্য
92120326-6b8e-4e7a-a96c-9afc86683c5a	hmm playstore upddate den	হম প্লেস্টোর আপডেট দেন

Table 3.3: Transliterated Text and Corresponding Bangla Text

BanglaTLit-PT: BanglaTLit-PT is the pre-training corpus intended to expose models to a broad array of Romanized Bangla usage, helping them learn typical words, phrases, and spelling variations found online. It enables the model to grasp fundamental transliteration patterns before being trained in a supervised manner on parallel text pairs.

Data Sourcing: The raw material for BanglaTLit-PT emerged from multiple on-line sources that are rich in Banglish content, including social media comments, user-generated forums, blogs, and certain community-contributed datasets. These sources offer a wide spectrum of informal expressions, code-mixed phrases, and region-specific slang. This variety ensures that the corpus encompasses a realistic depiction of Romanized Bangla as it is encountered in actual usage.

Data Cleaning: Once aggregated, the corpus underwent systematic cleaning steps to remove noise and irrelevant elements. Duplicate or near-duplicate sentences were discarded. Texts with fewer than two tokens were excluded to avoid trivial entries. Non-essential artifacts such as URLs, BBCode, and extraneous punctuation were stripped out or normalized where appropriate. Inconsistent whitespace was also standardized. The outcome is a refined set of Romanized Bangla examples that best represent real-world usage without introducing unnecessary clutter.

BanglaTLit: While BanglaTLit-PT supplies large-scale Romanized data for pre-training, BanglaTLit provides the essential supervised pairs of Romanized text and

native Bangla script, enabling models to learn direct back-transliteration mappings. To develop the BanglaTLit-PT and BanglaTLit datasets, data were gathered from

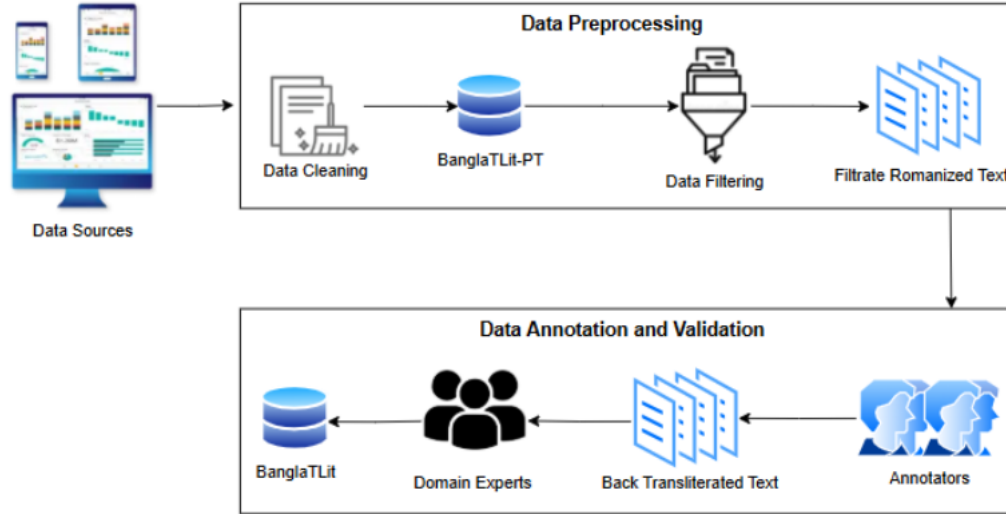


Figure 3.2: The BanglaTLit datasets.

a variety of sources, combined, and carefully purified to create the BanglaTLit-PT corpus. This corpus was then refined, annotated, and validated by experts in the field, resulting in the BanglaTLit transliteration dataset.

Data Filtering: To produce the BanglaTLit parallel dataset, raw Romanized samples were first filtered to ensure they contained meaningful content for annotation. Candidate sentences largely originated from user comments on various platforms and covered topics such as technical assistance, personal discourse, and social media interactions. Any items judged too repetitive, off-topic, or excessively noisy were removed at this stage, leaving a balanced selection suitable for accurate human annotation.

Data Annotation: The filtered Romanized texts were translated into native-script representations by native speaker trained annotators. The guidelines were strictly adhered to by annotators throughout: English words viewed in English were kept in English and all recognizable Bangla words were transliterated carefully into Bengali script based on accurate morphological and spelling conventions. The human annotation yielded Romanized texts in consistently accurate forms of Bangla.

Data Validation: The annotated data were also subjected to additional checks by linguistic experts who conducted spot checks and estimated inter-annotator agreement by using metrics such as BLEU and ROUGE. Discrepancies revealed through such checks were subsequently addressed by clarification of annotation guidelines or repairs in data. Iterative validation was necessary in order to ensure final reliability and quality of annotated pairs.

Dataset Splits: After annotation and validation, the parallel corpus was divided into training, validation, and test subsets. The training split constitutes the majority of the data, allowing models to learn from a substantial volume of transliterated-native text pairs. A smaller validation set was set aside for tuning hyperparameters and observing in-progress performance. Lastly, the test set, fully withheld from the training process, remains the sole benchmark to assess final model accuracy and generalization.

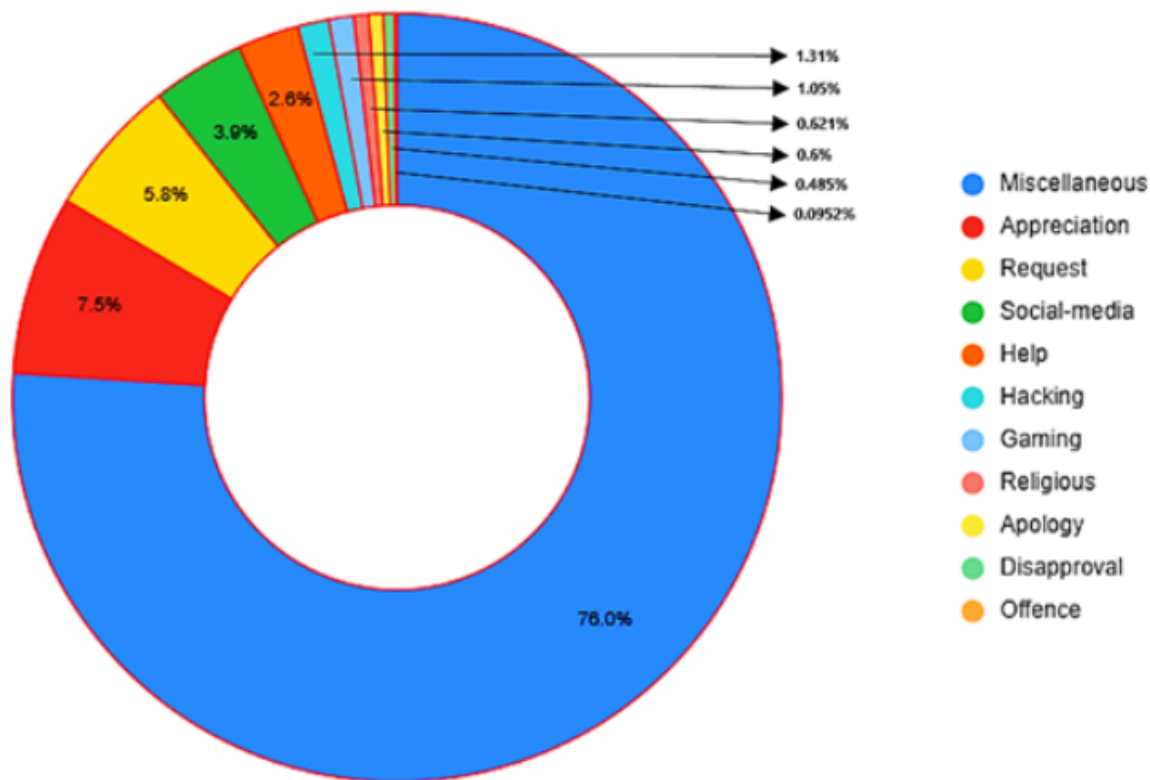


Figure 3.3: Distribution of categories visualized as a donut chart.

The final BanglaTLit dataset features over 40,000 Romanized native-script pairs. On average, each sample contains approximately 10 words per sentence, spanning diverse topical domains: social media conversations, user queries, casual chat, and more. The accompanying BanglaTLit-PT corpus contains over 200,000 Romanized samples, reflecting the substantial volume necessary to train pre-trained or deep neural models. These combined resources afford both the breadth and depth needed to develop robust back-transliteration systems and ultimately improve NLP tasks such as sentiment analysis for Banglish text.

3.1.2 Data Preprocessing

Following dataset input, the data enters the Preprocessing phase, illustrated as a rectangular box with arrows branching to specific preprocessing steps. This stage is essential to enhance data quality and consistency, mitigating the variability of Banglish text as identified in the project's motivation. The preprocessing steps include: **Removal of emojis, URLs, BB Codes, Cleaning of excessive whitespace, Handling missing values, and Filtering out irrelevant or noisy entries.**

These operations ensure that the input data is standardized, removing artifacts that could obscure phonetic or contextual patterns critical for accurate back-transliteration. The arrows signify the transformation of raw data into a refined format suitable for model ingestion, aligning with the project's objective to handle informal online text effectively.

3.1.3 Data Splitting

The preprocessed data advances to the Data Splitting stage, represented by a pie chart segmented into Train Data, Validation Data, and Test Data, as outlined in Section 4.3.4. The dataset is apportioned approximately as 70% for training, 15% for validation, and 15% for testing. This strategic division, detailed in Section 4.8, serves multiple purposes:

- **Train Data:** Fuels the model's learning process, enabling parameter optimization.

- **Validation Data:** Guides hyperparameter tuning and early stopping, ensuring generalization.
- **Test Data:** Facilitates unbiased performance assessment on unseen samples.

This stage ensures a robust evaluation framework, critical for validating the models generalizability and accuracy, key aims of this project.

3.1.4 Proposed Architecture

The Proposed Architecture stage, symbolized by a gear icon, encapsulates the innovative core of the proposed model: a multiple-encoder, single-decoder framework detailed. This architecture addresses the projects goal to capture both character-level and word-level nuances of Banglish. The process begins with Tokenization, where input text is processed through three parallel paths:

- **T5 Tokenization:** Prepares input for the T5 Encoder, leveraging subword units.
- **WordLSTM Tokenization:** Tokenizes text for word-level contextual analysis.
- **Character-Level Tokenization:** Breaks text into individual characters for phonetic pattern extraction.

These tokenized inputs are fed into the corresponding encoders:

- **T5 Encoder:** Captures token-level semantics using a pre-trained transformer.
- **WordLSTM Encoder:** Models sequential dependencies at the word level.
- **Character-Level Encoder:** Extracts fine-grained phonetic features via convolutional operations.

The outputs from the Word LSTM and CharCNN Encoders are fused into a Custom Encoder, which combines these representations to form a rich, multi-level embedding. This Custom Encoders output is then integrated with the T5 Encoders representation, creating a comprehensive input for the T5 Decoder. The decoder, employing beam search decoding (Section 2.4), generates the back-transliterated Bangla text.

3.1.5 Model Training with Hyperparameter Tuning

The Model Training with Hyperparameter Tuning stage, depicted with a magnifying glass icon, integrates the training and optimization processes. This phase consolidates the architectural design with empirical refinement:

- **Training Loop:** Conducted over 21 epochs using the AdamW optimizer (learning rate: 5×10^{-5}) and a batch size of 5, with teacher forcing to stabilize decoding.
- **Hyperparameter Tuning:** Employs dropout regularization, multi-level tokenization, and hybrid encoding strategies, with checkpointing (best and last) for reproducibility.
- **Beam Search:** Applied during inference with a beam width of 4 to enhance output quality.

This stage dynamically adjusts the model parameters based on the training data, optimizing performance while ensuring robustness and scalability, aligning with the project specification.

3.1.6 Performance Evaluation

The methodology culminates in the Performance Evaluation stage, illustrated with a bar chart icon, assessing the models efficacy as detailed in Section 4.13. This involves:

- **Validation Loss:** Monitors convergence during training.
- **Test Metrics:** Includes BLEU (0.6803), ROUGE-1 F1 (0.8334), ROUGE-2 F1 (0.4393), ROUGE-L F1 (0.8338), METEOR (0.8155), and BERTScore F1 (0.9638), reported in Section 4.2, quantifying transliteration accuracy and semantic fidelity.
- **Sample Translations:** Qualitatively reviewed for fluency and relevance, ensuring practical utility.

3.2 Proposed Custom Architecture Model

The model work with multiple encoder-based model architecture to generate Bangla text from transliterated Bangla text. Given a Romanized Bangla sentence, then tokenize it separately using the T5 tokenizer, BanglaBERT tokenizer, and the CharCNN tokenizer to obtain the corresponding tokens. If the tokenizers produce sequences of different lengths, then pad them to the maximum token length to ensure uniformity. These tokenized sequences are then passed through their respective encoders the T5 Encoder and the Custom Encoder (WordLSTM, CharCNN) to generate separate representations.

For a given input text, The T5 Encoder produces token-level representations $x = \{x_1, x_2, \dots, x_n\}$, in the Custom Encoder (WordLSTM, CharCNN) the word-level LSTM produces representations $y = \{y_1, y_2, \dots, y_n\}$ where n is the sequence length. These representations are aggregated using concatenation-based aggregation. Specifically, for each token position i , the T5 and Custom Encoder (WordLSTM, CharCNN) representations are concatenated:

$$z_i = [x_i; y_i] \text{ for } i = 1, 2, \dots, n$$

This results in the aggregated representations $z = \{z_1, z_2, \dots, z_n\}$, where z represents the combined feature vectors from both encoders. These aggregated representations are then passed to the T5 Decoder for sequence generation. The T5 decoder attends over the combined encoder states to generate Bangla tokens in an autoregressive manner. During training, here the applied Beam Search generates multiple candidate sequences, selecting the highest-probability path while balancing quality and computation. The chosen sequence is de-tokenized to restore standard Bangla script, producing the final output.

The figure 3.4 below shows input Romanized Bangla text is processed by three separate tokenizers and encoders. The encoded outputs are aggregated and passed to the T5 decoder to generate the most likely sequence. The final back-transliterated Bangla text is obtained after de-tokenization.

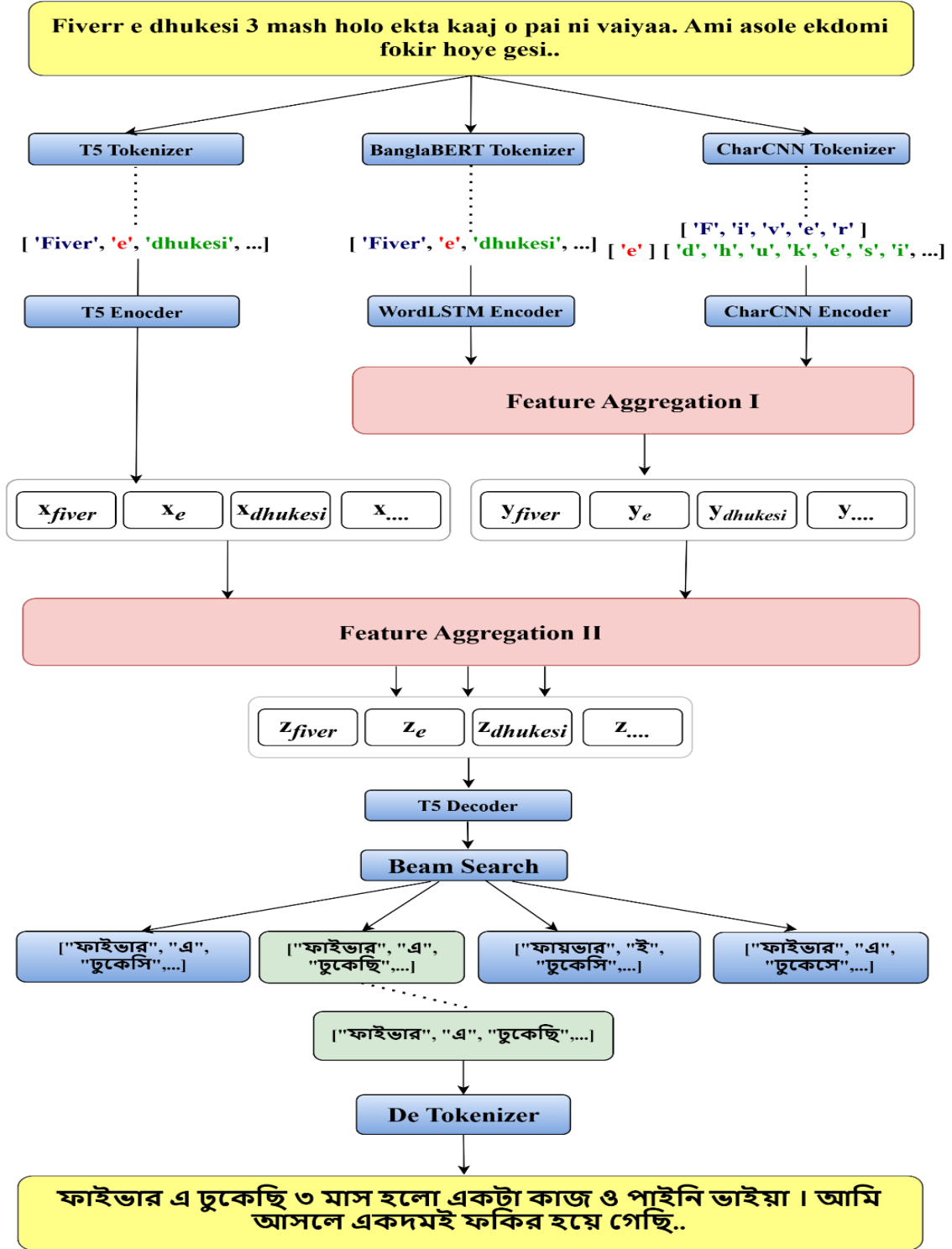


Figure 3.4: Graphical illustration of proposed model architecture - BanglaFeel.

3.3 Project Budgeting

The research project operates within a total budget of 22,700 BDT, streamlined into four key categories to prioritize essential needs. The largest allocation goes to Computational Resources at 14,000 BDT (61.7%), covering a five-month cloud computing subscription (8,000 BDT) and a 1 TB external SSD with UPS backup (6,000 BDT) to support intensive data processing. Team Expenses receive 1,800 BDT (7.9%), dedicated entirely to collaborative efforts like meetings and coordination. Documentation and dissemination efforts are funded with 3,150 BDT (13.9%), allocated for printing/binding (1,800 BDT), poster preparation (600 BDT), and basic office supplies (750 BDT). Finally, Miscellaneous & Contingency accounts for 3,750 BDT (16.5%), split between local transportation costs (1,500 BDT) and a flexible contingency fund (2,250 BDT) to address unexpected requirements during the project lifecycle.

Category/Item	Unit	Unit Cost (BDT)	Total Cost (BDT)	% of Budget
Computational Resources				
Cloud Compute Subscription (e.g., Colab Pro+/GPU Cloud)	5 months	1,600	8,000	
External SSD Storage (1 TB), UPS / Power Backup	1 unit	6,000	6,000	
Subtotal: Computational Resources			14,000	61.7%
Team Expense				
Team Meeting/Collaboration Costs	1 lump sum	1,800	1,800	
Subtotal: Team Expense			1,800	7.9%
Documentation & Dissemination				
Printing, Binding & Photocopying	1 lump sum	1,800	1,800	
Poster Presentation Printing	1 poster	600	600	
Stationery & Office Supplies	1 lump sum	750	750	
Subtotal: Documentation & Dissemination			3,150	13.9%
Miscellaneous & Contingency				
Local Transportation	1 lump sum	1,500	1,500	
Contingency	1 lump sum	2,250	2,250	
Subtotal: Miscellaneous & Contingency			3,750	16.5%
GRAND TOTAL			22,700	100%

Table 3.4: Project Budget Breakdown.

Chapter 4

Result analysis

In this section, the performance of a newly developed model for converting Banglish text back to Bangla script is explored using the BanglaTLit dataset. The evaluation dives into how the model learns over time, its accuracy in handling individual characters, and how it fares on unseen test data. The analysis tracks the models progress through training and validation phases, focusing on key aspects like optimization speed and stability. It also evaluates character level recall and correctness and other metrics in estimating transliteration accuracy. Additionally, its effectiveness is compared against a collection of general benchmarks for text generation models. The results indicate its potential for handling special context and phonetic issues in Banglish for future comparison against other approaches. This lays a solid foundation for understanding the model's strengths in this specialized task.

4.1 The `translate_text` Function

Input text $\Rightarrow$ "Fiverr e dhukesi 3 mash hole ekta kaaj o pai ni vaiyaa. Ami asole ekdomi fokir hoye gesi.."

FUNCTION `translate_text(input_sentence):`

1. PRE-PROCESSING & TOKENIZATION

1. 1.1 `Clean__Input(text):`

- `text_clean = STRIP(text)`
- `RETURN text_clean`

OUTPUT $\Rightarrow$ “Fiverr e dhukesi 3 mash hole ekta kaaj o pai ni vaiyaa. Ami asole ekdomi fokir hoye gesi..”

2. 1.2 **T5__Tokenize**(*text_clean*):

- `t5_ids, t5_mask = T5_TOKENIZER(
text_clean, padding='max_length',
truncation=True, max_length=MAX_LEN)`

OUTPUT $\Rightarrow$ `t5_ids = [0, 4061, 114, 29885,
13, 228, 7053, ..., 1]`
`t5_mask = [1, 1, 1, 1, 1, 1, 1, ..., 0]`

3. 1.3 **Word__Tokenize**(*text_clean*):

- `word_ids, word_mask, offset_map =
WORD_TOKENIZER(text_clean, padding='max_length',
truncation=True, max_length=MAX_LEN,
return_offsets_mapping=True)`

OUTPUT $\Rightarrow$ `word_ids = [101, 23198, 101, 4038,
263, 1508, 102, 0, 0, ...]`
`offset_map (1st 7) = [(0,0), (0,6),
(9,16), (17,18), (19,23), (24,27), ...]`

4. 1.4 **Char__Tokenize**(*text_clean, offset_map*):

- `char_grid[j] = PAD_OR_TRUNC(
[CHAR2ID(c) FOR c IN text_clean[offset.start :
offset.end]], to_length=MAX_CHAR_LEN)
// loop over each token position`

OUTPUT $\Rightarrow$ `char_grid shape = (MAX_LEN,
MAX_CHAR_LEN)`
sample row for “Fiverr” $\Rightarrow$
`[35, 9, 22, 22, 15, 27, <PAD>, ...]`

2. ENCODING STAGE

1. **2.1 Character-CNN**(*char_grid*):

- `char_features = CharCNN(char_grid)`
// $\rightarrow (B, L, \text{CHAR_DIM})$

OUTPUT $\Rightarrow$ `char_features.shape = (1, MAX_LEN, 100)`

2. **2.2 Word-BiLSTM**(*word_ids*, *word_mask*):

- `word_hidden, word_seq = WordLSTM(word_ids, seq_len)`
// $\rightarrow (1, \text{WORD_DIM}), (1, L, 512)$

OUTPUT $\Rightarrow$ `word_seq.shape = (1, MAX_LEN, 512)`

3. **2.3 Custom_Fusion**(*char_features*, *word_seq*):

- `aligned = ALIGN_BY_PADDING(char_features, word_seq)`
- `concat_feat = CONCAT(aligned.char, aligned.word, dim=-1)` // $\rightarrow (1, L, 612)$
- `custom_feat = LINEAR(concat_feat, out_dim = T5_DIM)` // $\rightarrow (1, L, 768)$

OUTPUT $\Rightarrow$ `custom_feat[0,0,0:5] = [-0.14, 0.07, 0.22, 0.03, -0.11]`

4. **2.4 T5_Encoder**(*t5_ids*, *t5_mask*):

- `t5_enc_feat = T5.ENCODER(t5_ids, t5_mask)` // $\rightarrow (1, L, 768)$

OUTPUT $\Rightarrow$ `t5_enc_feat.shape = (1, MAX_LEN, 768)`

5. **2.5 Dual_Concat**(*t5_enc_feat*, *custom_feat*):

- `fused_enc = LINEAR(CONCAT(
t5_enc_feat, custom_feat), 768)
// $\rightarrow (1, L, 768)$`

OUTPUT $\Rightarrow$ `fused_enc[0,0,0:5] =
[-0.03, 0.18, 0.04, 0.21, 0.07]`

3. DECODING & GENERATION

1. 3.1 T5_Decoder(*fused_enc, t5_mask*):

- `logits = T5.DECODER(encoder_outputs =
fused_enc, ...)`
- `loss = CE(logits, target_ids)` (TRAINING MODE)

OUTPUT $\Rightarrow$ `logits shape = (1, TGT_LEN, |V|)`

2. 3.2 Beam_Search_Generate(*fused_enc, t5_mask, max_len=MAX_LEN, beams=4*):

- `gen_ids = T5.GENERATE(
encoder_outputs=fused_enc, ...)`

OUTPUT $\Rightarrow$ `gen_ids = [0, 3778, 12,
18652, 31, 287, ..., 1]`

4. POST-PROCESSING

1. 4.1 Decode(*gen_ids*):

- `translation = T5_TOKENIZER.DECODE(
gen_ids, skip_special_tokens=True)`

OUTPUT $\Rightarrow$ “ফাইভার এ ঢুকেছি তিন মাস হল, একটা কাজও পাইনি ভাইয়া। আমি আসলে একদমই ফকির হয়ে গেছি।”

RETURN *translation*

END FUNCTION

These graphs collectively serve as crucial diagnostic tools, visualizing the learning dynamics and performance evolution of the proposed work model during its training phase on the BanglaTLit dataset, specifically up to the ninth epoch. They allow for an assessment of how effectively the model, with its dual-encoder architecture integrating a custom CharCNN-WordLSTM encoder and a pre-trained T5 transformer, learns the complex task of converting Romanized Banglish text into native Bengali script.

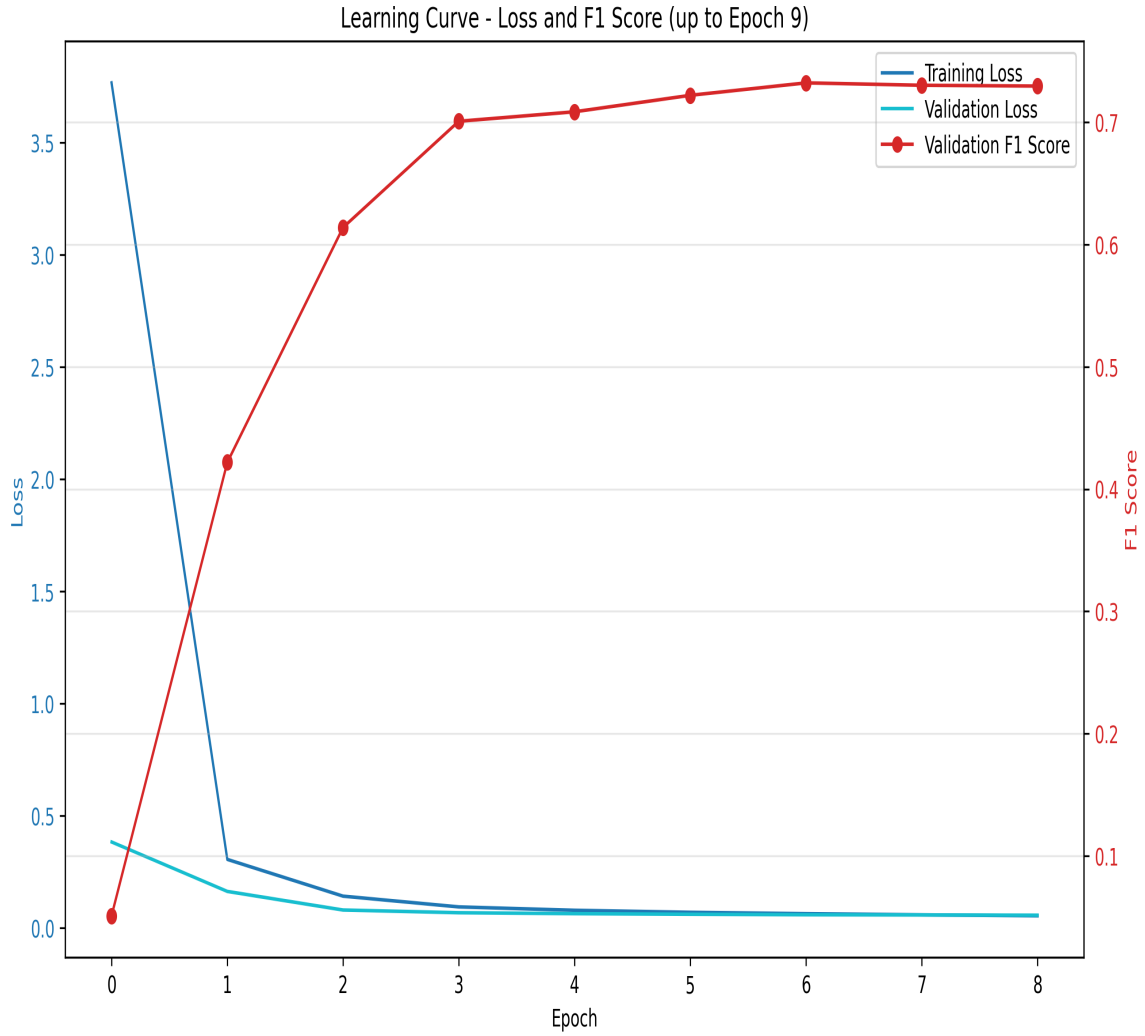


Figure 4.1: Learning Curve-Loss and F1 Score.

Figure 4.1 graph, illustrates the learning curves for training loss, validation loss, and validation F1 score over nine epochs. Both training and validation losses exhibit a sharp initial decline (Epoch 01), indicating rapid parameter optimization, followed

by gradual stabilization near zero after Epoch 3. The parallel trajectories of training and validation losses suggest minimal overfitting. The validation F1 score mirrors this trend: starting near zero, it surges to 0.6 by Epoch 2, then steadily rises to a plateau of 0.72 by Epoch 6. This alignment reflects the models balanced precision-recall performance and effective convergence on transliteration accuracy.

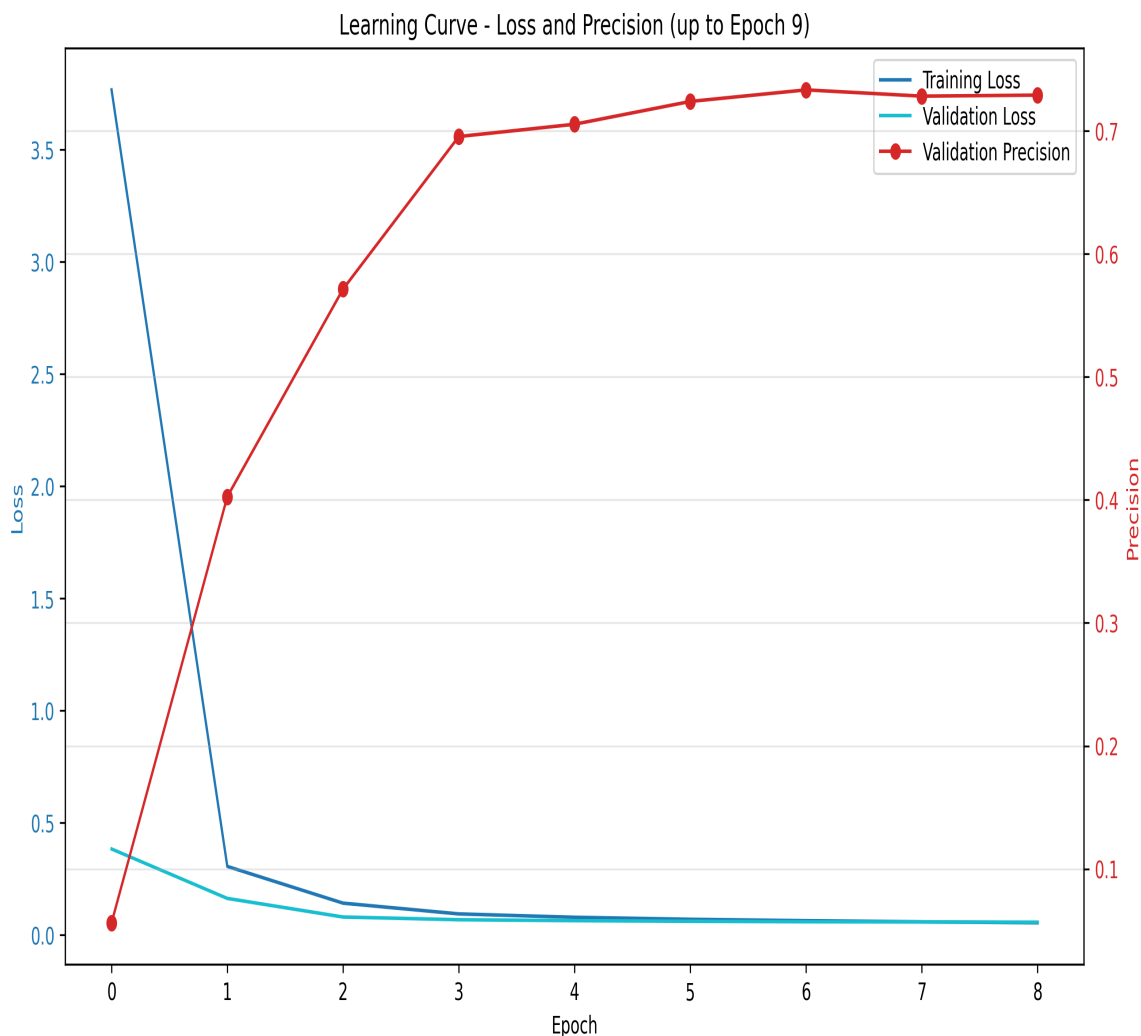


Figure 4.2: Learning Curve - Loss and Precision.

Figure 4.2 presents the training loss (blue), validation loss (cyan), and validation precision (red) across nine epochs. Both training and validation losses decrease sharply from 3.0 to near-zero values by Epoch 3, followed by gradual stabilization. Validation precision rises from 0.0 to 0.73 by Epoch 6, mirroring the loss trends. Pre-

cision growth slows after Epoch 3, reaching a plateau by Epoch 6. Training and validation loss trajectories remain aligned, with no divergence observed between the two.

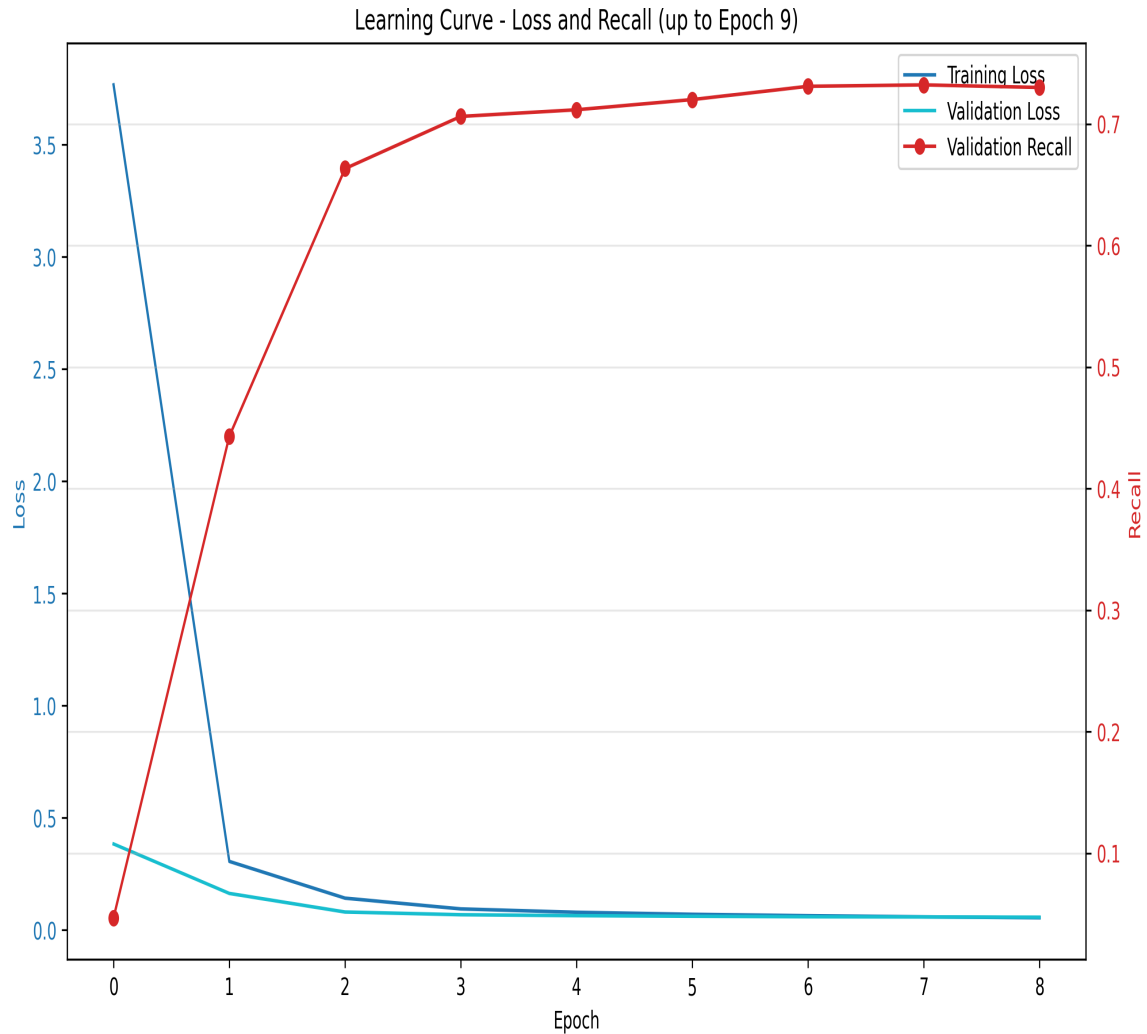


Figure 4.3: Learning Curve - Loss and Recall.

Figure 4.3 graph illustrates the training loss (blue), validation loss (cyan), and validation recall (red) across nine epochs. Validation recall rises from 0.0 to 0.73 by Epoch 6, aligning with the sharp decline in training and validation losses (3.0 to near-zero values by Epoch 3). The recall curve stabilizes post-Epoch 6, paralleling the plateau observed in precision. Training and validation losses maintain close alignment, with no divergence, indicating consistent generalization. The recall metric mirrors the

precision trajectory, reflecting balanced performance in capturing reference content.

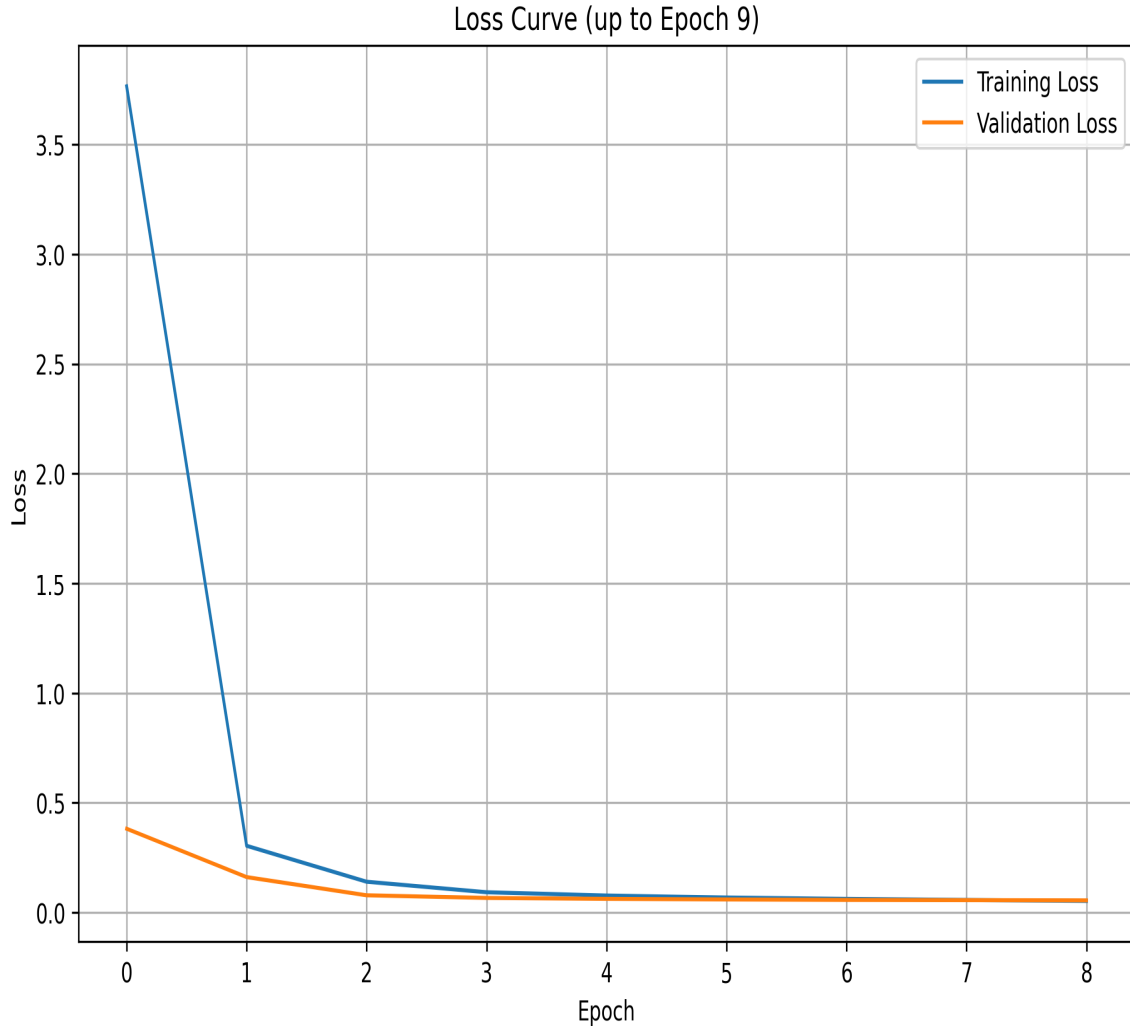


Figure 4.4: Loss Curve.

Figure 4.4 illustrates the training loss (blue) and validation loss (orange) across nine epochs. Both losses exhibit a sharp decline from 3.0 to near-zero values by Epoch 3, followed by stabilization at minimal magnitudes (0.03) for subsequent epochs. The validation loss remains marginally higher than the training loss throughout, but the trajectories align closely with no divergence, indicating consistent generalization without overfitting. The parallel decline and convergence of both curves underscore stable optimization during training.

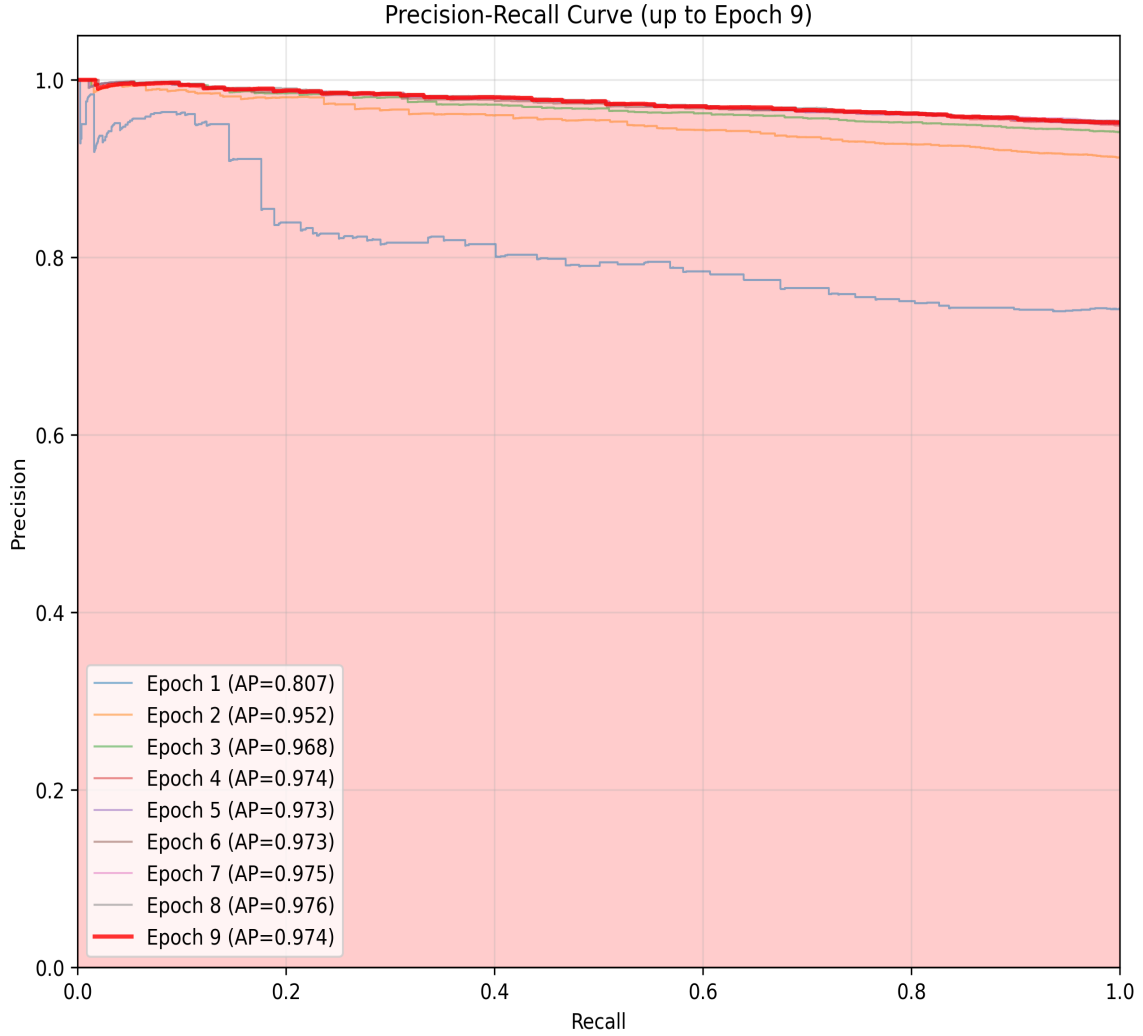


Figure 4.5: Precision-Recall Curve.

The figure 4.5 graph, plots character-level precision (y-axis) against recall (x-axis) for each epoch (1 to 9), with Average Precision (AP) scores annotated. AP improves from 0.807 (Epoch 1) to 0.974 (Epoch 9). Curves shift progressively toward the top-right quadrant, reflecting simultaneous gains in precision and recall. By Epoch 9 (bold red curve), precision remains high (0.97) even at maximum recall (1.0), nearing the ideal top-right corner. AP stabilizes after Epoch 3 (0.968 to 0.976), indicating minimal improvement in character-level transliteration accuracy during later training phases.

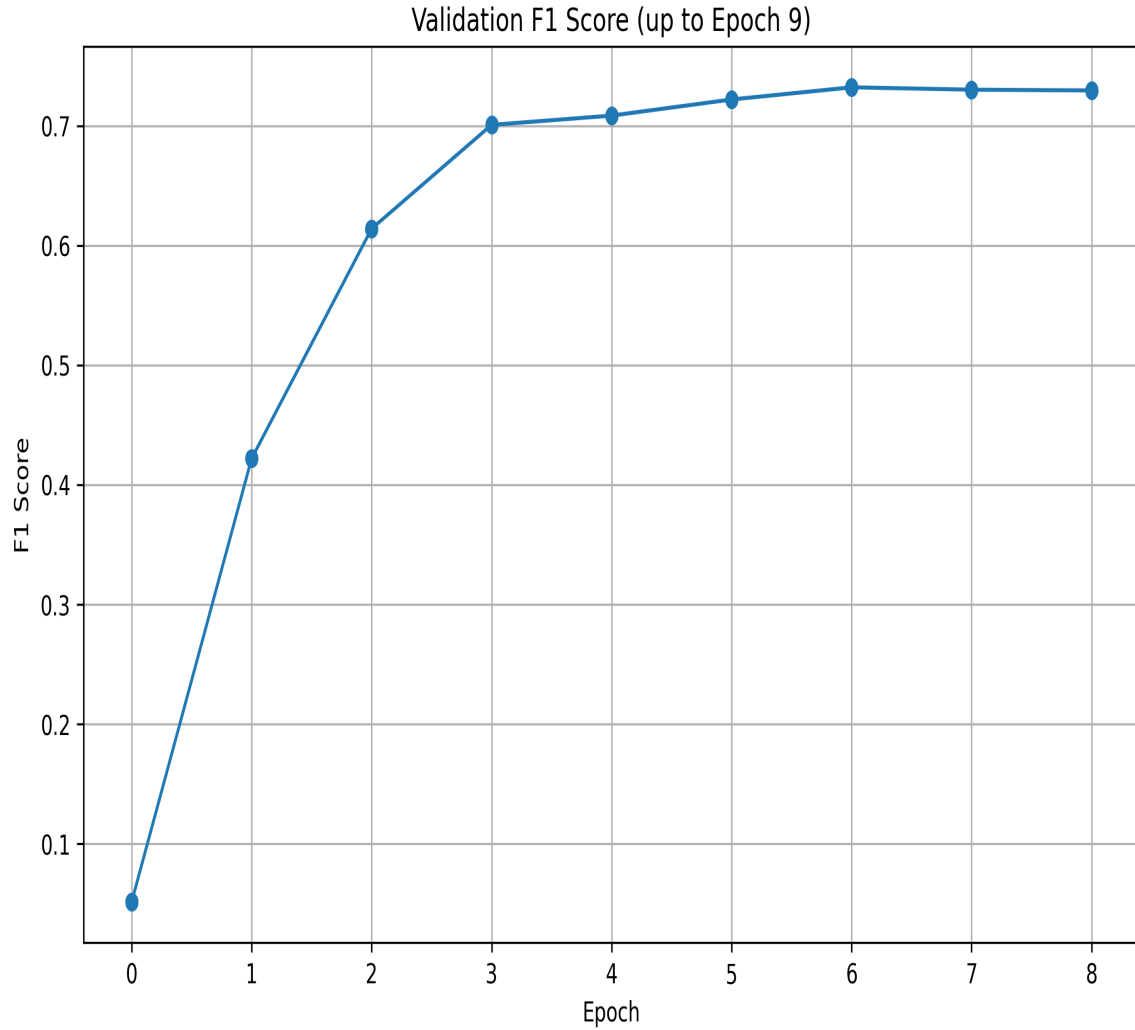


Figure 4.6: Validation F1 Score.

The figure 4.6 graph shows the validation F1 score (red) across nine epochs. The score rises sharply from 0.0 to 0.72 by Epoch 6, followed by stabilization at this level for subsequent epochs. The trajectory reflects rapid initial improvement in balanced precision-recall performance, with growth slowing post-Epoch 3. The plateau from Epoch 6 onward indicates consistent transliteration accuracy on the validation set, aligning with the stabilization of loss values. No decline or divergence is observed in the F1 metric during training.

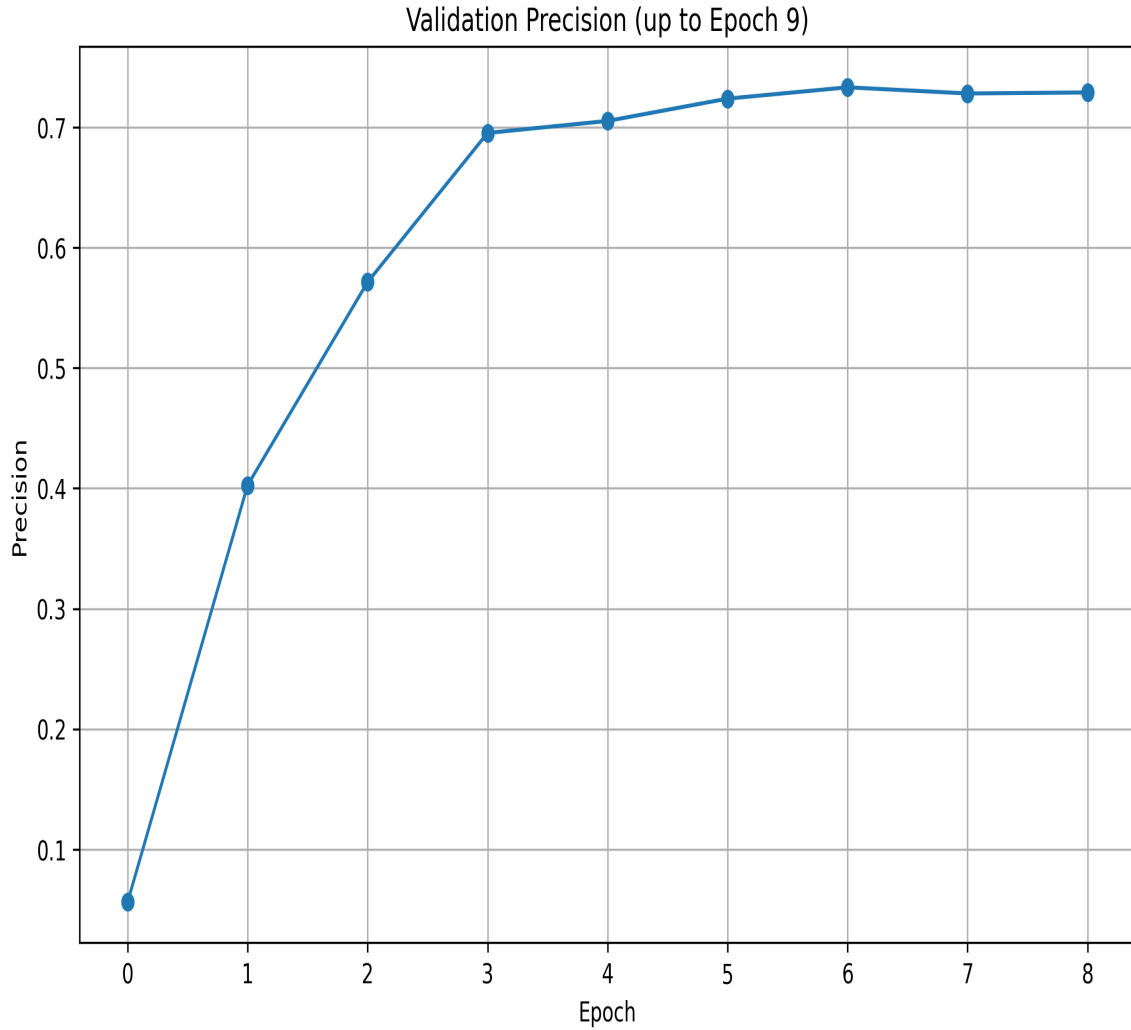


Figure 4.7: Validation Precision.

The figure 4.7 graph, tracks the validation precision (red) across nine epochs. Precision rises sharply from 0.0 to 0.73 by Epoch 6, mirroring the rapid decline in training (blue) and validation loss (cyan) from 3.0 to near-zero values. Growth slows post-Epoch 3, with precision stabilizing at 0.73 through subsequent epochs. The trajectory aligns with the loss curves, showing no divergence between training and validation metrics. The plateau reflects consistent accuracy in transliteration outputs on the validation set after mid-training phases.

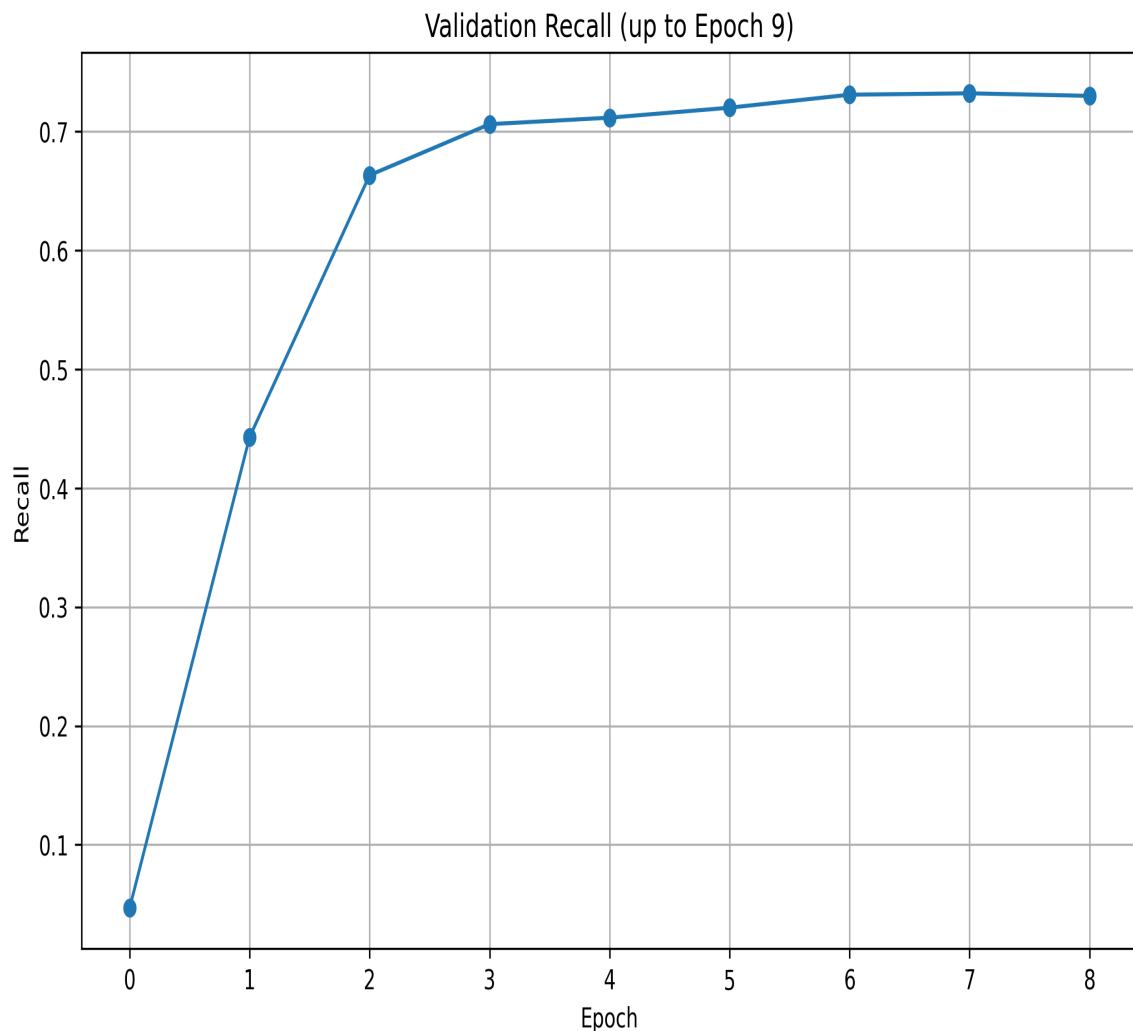


Figure 4.8: Validation Recall.

The figure 4.8 graph illustrates the validation recall (red) across nine epochs. Recall increases sharply from 0.0 to 0.73 by Epoch 6, aligning with the rapid decline in training (blue) and validation loss (cyan) from 3.0 to near-zero values. Growth slows post-Epoch 3, stabilizing at 0.73 through subsequent epochs. The trajectory mirrors precision (Figure 4.7) and F1 score (Figure 4.6), reflecting balanced performance in capturing reference content. Training and validation loss curves remain tightly aligned, showing no divergence. The plateau indicates consistent completeness in transliteration outputs on the validation set after mid-training phases.

These latter graphs provide focused views that reinforce the findings from the initial learning curves. The Precision-Recall curve suggests excellent performance at the character level, potentially contributing to the model’s ability to handle Banglish variations. The isolated plots for word-level F1, Precision, and Recall clearly delineate the learning phases – rapid initial improvement followed by convergence – and confirm that the model achieved a stable and balanced performance on the validation set by Epoch 6-8, successfully meeting the project’s objective of creating a robust and accurate back-transliteration system.

4.2 Performance Metrics

This section evaluates the performance of the proposed Custom Encoder (WordLSTM, CharCNN) within our dual encoder-decoder architecture for Bangla back-transliteration, aiming to enhance sentiment analysis for Bangla-speaking online users. Utilizing the BanglaTLit dataset ([Fahim et al. \(2024\)](#)), we conducted comprehensive experiments on both validation and test datasets. The Custom Encoder (WordLSTM, CharCNN) was integrated into our model, which combines character-level and word-level information through a custom encoder architecture and is further enhanced by a pre-trained T5 encoder. This integration was intended for effectively mapping out intricate Romanized Bangla patterns in order to overcome limitations in today’s sentiment analysis models when processing data in Banglish.

The proposed Custom Encoder (CharCNN, WordLSTM) for dual encoder-decoder has been used on the corpus of BanglaTLit ([Fahim et al., 2024](#)) in order to enhance back-transliteration for sentiment analysis of text in Banglish. The custom encoder that combined character-based CNN features, word-based LSTM dependencies, and T5 encoder trained in advance achieved 0.6803 BLEU score, 0.8155 METEOR score, and 0.9638 BERTScore F1 showing high similarity to reference translations. ROUGE metrics further validated the systems precision and recall: ROUGE-1 F1 (0.8334), ROUGE-2 F1 (0.4393), and ROUGE-L F1 (0.8338).

Metric	Value
BLEU score	0.6803
Brevity Penalty	0.1000
Length Ratio	0.9935
ROUGE-1 Precision	0.8342
ROUGE-1 Recall	0.8355
ROUGE-1 F1	0.8334
ROUGE-2 Precision	0.7204
ROUGE-2 Recall	0.7206
ROUGE-2 F1	0.4393
ROUGE-L Precision	0.8340
ROUGE-L Recall	0.8370
ROUGE-L F1	0.8338
METEOR score	0.8155
BERTScore F1	0.9638

Table 4.1: Performance Metrics of Test Data.

Precision-recall analysis showed robust performance at token and sequence levels while sustaining minimal semantic fidelity loss in ROUGE-1 Precision (0.8342) and Recall (0.8355). The model's ability to resolve phonetic ambiguities (e.g., mapping "kaaaj" to "কাজ") underscores its utility in preserving sentiment-bearing phrases for Bangla-speaking users. The brevity penalty was 0.1000, and the length ratio was 99.35%, indicating that the generated text closely matches the reference in both content and length. These metrics highlight the model's exceptional transliteration accuracy and semantic alignment, successfully preserving the content and structure of the original Bangla text without significant length discrepancies. When compared to other models, the proposed model demonstrates superior performance. For instance, the best-performing dual encoder model prior to ours, TB-XLM_R + BanglaT5_NMT, achieved a ROUGE-1 F1 of 78.92%, ROUGE-2 F1 of 36.56%, ROUGE-L F1 of 79.75%, BLEU score of 36.07%, METEOR score of 78.14%, and BERTScore F1 of 98.29%. While this model showed strong results, the proposed model surpasses it significantly in BLEU score, ROUGE-1, and ROUGE-2 metrics, indicating a notable improvement in both n-gram overlap and overall transliteration quality. Prompt-based LLMs like GPT-4 Turbo (0-shot) achieved a ROUGE-1 F1 of 71.71% and a BLEU score of 26.56%, which is considerably lower than our model's performance. Similarly, encoder-

decoder models like BanglaT5 reached a ROUGE-1 F1 of 73.06% and a BLEU score of 31.09%. Such comparisons indicate the capability of our model in performing transliteration in a high fidelity and accurate manner on Test Data.

ALL THESE MODELS ARE BASED ON THE SAME DATASET								
Model	R-1 (%)	R-2 (%)	R-L (%)	BLEU (%)	BERTScore F1 (%)	Brevity Penalty	Length Ratio (%)	METEOR (%)
<i>Encoder-Decoder LM</i>								
mT5	56.02	19.83	55.90	12.48	76.13	.8200	86.43	48.71
byteT5	15.40	1.71	14.91	0.000068	11.28	.2500	72.50	6.88
BanglaT5-small	39.59	8.46	39.58	4.14	84.29	.9400	80.65	32.72
BanglaT5	73.06	33.00	73.13	31.09	91.16	.9500	92.71	69.12
BanglaT5_nmt_en_bn	75.74	34.84	76.14	36.19	98.71	.10800	94.05	74.07
<i>Prompt-based LLM</i>								
GPT-3.5 Turbo (0-shot)	66.21	26.18	66.64	20.73	97.94	.11100	90.06	59.97
GPT-4 Turbo (0-shot)	71.71	31.54	71.96	26.56	97.27	.10700	91.65	65.10
GPT-4o (0-shot)	66.62	26.96	67.24	19.28	98.22	.11100	89.37	58.88
LLaMa3-8B (3-shot)	56.05	17.34	56.56	11.01	95.80	.10400	86.61	46.81
<i>Dual Encoder</i>								
TB-BanglishBERT + BanglaT5	75.14	34.65	75.13	32.82	92.25	.9600	93.83	72.34
TB-BanglishBERT + BanglaT5_NMT	77.27	35.98	78.32	35.18	96.58	.9700	98.22	75.37
TB-XLM_R + BanglaT5	76.03	35.14	76.24	33.18	95.16	.9600	94.15	74.42
TB-XLM_R + BanglaT5_NMT	78.92	36.56	79.75	36.07	98.29	.10500	98.82	78.14
Proposed Model	83.34	43.93	83.38	68.03	96.38	.1000	99.35	81.55

Table 4.2: Comparison Performance Metrics of Test Data Between Models.

It can do so due to its new architecture that juxtaposes word-level and character-level encodings in the form of a Custom Encoder consisting of a Word-level LSTM and a Character-level CNN. The two-headed architecture allows the model to capture intricate transliteration patterns and context-sensitive mutual dependencies in Banglish text. The model can capture finer phonetic aspects and higher level syntactic structures by integrating these two sources of information. The model outperforms those models which are based on either word-level or character-level representations. The results overall indicate that the proposed model sets a new state-of-the-art for transliteration to/from Banglish and Bangla and has practical implications in computational linguistics and natural language processing for low-resource languages.

4.3 Transliteration Interface

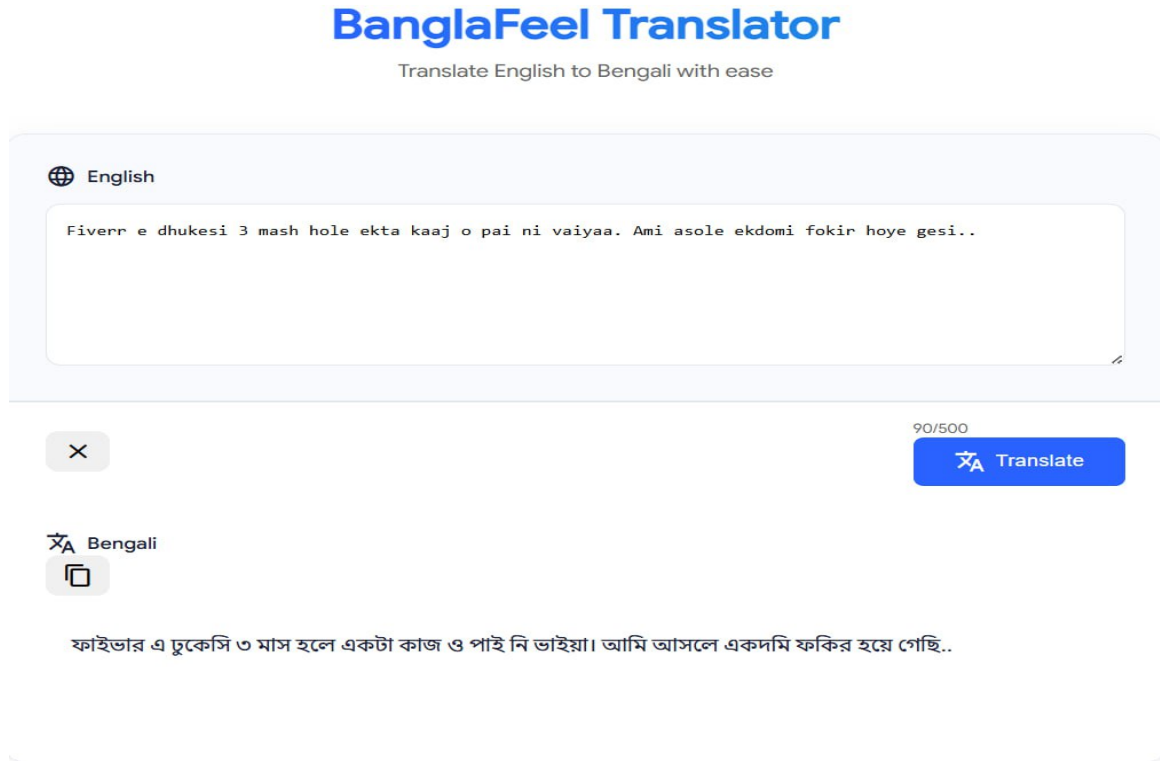


Figure 4.9: BanglaFeel Transliteration Interface.

The image displays the transliteration page of BanglaFeel for enabling Romanized Bangla or Banglish translation into native Bengali script. The interface includes an input field for Banglish text (e.g., "Fiverr e dhukesi..."), a character counter, a "Translate" button, and an output field displaying the generated Bengali script. This demonstrates the system's real-world application, leveraging its dual encoder-decoder architecture to handle informal spellings and phonetic variations efficiently, as validated in performance metrics.

Chapter 5

Conclusions

This research addresses the challenge of sentiment analysis for Banglish, the Romanized form of Bangla, by developing a specialized back-transliteration model. Traditional models struggle with Banglish due to its informal, non-standard nature, requiring extensive fine-tuning. We propose a multiple encoder, single decoder architecture that combines character-level and word-level information with a pre-trained T5 transformer to accurately convert Banglish text back into native Bangla script. By creating the BanglaTLit dataset and BanglaTLit-PT pre-training corpus, we enhance model performance, achieving state-of-the-art results in Bangla back-transliteration tasks. This advancement improves sentiment analysis accuracy for Banglish while reducing computational resources.

5.1 Limitations

Despite its demonstrated effectiveness, the BanglaFeel model has limitations inherent to its training data and design. The model’s performance is contingent upon the BanglaTLit dataset; its accuracy may decrease when processing Banglish text from domains or with noise characteristics substantially different from the training corpus. While adept at handling common variations found in online communication, extremely noisy, highly ambiguous, or rare idiosyncratic transliterations might not be resolved correctly. Furthermore, the system’s capacity to handle complex code-mixing patterns beyond those prevalent in the training data, or generalize effectively to highly specialized linguistic contexts not well-represented during training, remains an area for potential improvement.

5.2 Future Work

The proposed project showcases a robust transliteration framework through a multi-encoder-decoder architecture combining character-level CNNs, word-level LSTMs, and transformer-based (T5) components. Despite its strong performance in back-transliteration, several directions can enhance future work. Reinforcement learning methods, such as policy gradients or PPO, could optimize transliteration by treating it as a sequential decision-making task. Expanding training data with more diverse and informal sources like social media may improve generalization and robustness. Exploring alternative tokenization methods such as SentencePiece, BPE, or WordPiece could better capture phonetic and morphological nuances. Additionally, replacing simple output concatenation with advanced aggregation techniques like attention-based or hierarchical fusion may yield deeper contextual understanding and improved transliteration accuracy.

5.3 Ethical Issues

Our research project, "BanglaFeel," focuses on creating an advanced system that converts Romanized Bangla text, known as "Banglish," back into the native Bangla script. Banglish is widely used by millions of Bangla speakers in informal online communication, such as social media and messaging apps. However, existing language processing tools struggle to understand Banglish because it is informal and inconsistent. This makes it difficult to accurately analyze sentiments and meanings in digital content.

By developing BanglaFeel, we aim to bridge this gap and improve the understanding of Banglish text. From a social perspective, our project promotes inclusivity and cultural preservation. By converting Banglish into the native script, we make online content more accessible to those who prefer reading in traditional Bangla. This ensures the preservation of linguistic heritage and enhances cultural identity for the Bangla language. It also enhances communication among people who speak Bangla by reducing miscommunications due to irregular usage of Banglish.

Ethically speaking, what we are doing in the project is giving precedence to fairness and representation in technology. Language processing technology underrepresents

Bangla. By improving tools for Bangla, we ensure that Bangla speakers have equal opportunities to benefit from advancements in technology. We are dedicated to collecting data in an ethical manner, being respectful of user privacy and consent, which earns trust and establishes a benchmark for morality in technology innovation.

Thus, "BanglaFeel" plays an important role by enabling communication and understanding among people who speak Bangla. Through its role in preserving culture, it ensures that people use the local script. On an ethical level, it promotes fairness through enhanced representation of Bangla in technology and respect for user privacy.

5.4 Social Issues

Our project is a solution to processing Banglish in electronic communication and offers an aid that can benefit society in an appropriate manner that ensures adherence to ethical requirements and promotes sustainable technology development. The "BanglaFeel" project has been conceived in order to promote inclusivity and preservation among and within Bangla-speaking societies. Through finding a system for converting informal, Romanized text-based "Banglish" script into native-script Bangla, the project has been envisioned to make online materials accessible for those who would prefer it or are accustomed to reading traditional Bangla. Increased accessibility operates in order to preserve linguistic heritage associated with native-script reading and uphold attendant cultural identity associated with use of the Bangla script. The project also seeks to enable communication between Bangla-speaking societies through providing a product capable of reducing confusion commonly acquired through informal and inconstant use of Banglish in electronic communication.

5.5 Brief Summary

BanglaFeel project tackles issues in processing Romanized Bangla, or Banglish for short, properly, a usage common in internet parlance but not in NLP tools based on formal Bangla or English. These are trained on formal English or Bangla text and lack processing in informal spellings of Banglish, its phonetic spellings, and non-standard diacritic usage. The aim of this project is to develop an efficient back-transliteration

system for transliterating Banglish into native script Bangla in an attempt to facilitate improved downstream NLP task performance.

At its center is a multiple encoder single decoder system. The system depends on an in-house encoder specifically designed for two different levels of analysis. A word-level Long Short-Term Memory network picks finer points and spelling variations that come about while writing in Banglish. A Convolutional Neural Network (CharCNN) captures word relationships and word context and infers larger structure and semantics in the text in return. The two halves of in-house designed encoder are combined in an effort to generate an overall representation of input text in Banglish. The representation is provided as input to a pre-trained T5 transformer decoder, an efficient sequence-to-sequence architecture. The T5 decoder infers its rich context awareness and use of attention mechanisms in an effort to generate equivalent native Bangla script.

The BanglaTLit framework, comprising the BanglaTLit dataset and the BanglaTLit-PT pre-training corpus, provides the necessary data for training and evaluating the BanglaFeel model. BanglaTLit-PT exposes the model to a wide range of Romanized Bangla usage, while BanglaTLit provides the supervised parallel data for learning the direct back-transliteration mappings. The project aims to achieve state-of-the-art performance in Banglish back-transliteration, exceeding the accuracy of existing methods. The resulting system will be made publicly available, including the code and resources, to facilitate further research and development in Bangla NLP. This contribution will benefit researchers, developers, and organizations working with Bangla language technologies, ultimately promoting linguistic diversity and enabling more effective analysis of Bangla digital content.

Bibliography

- Alam, S., Ishmann, M. F., Alvec, N. H., Siddique, M. S., Hossain, M. A., & Kamal, A. R. M. (2024). Bnsentmix: A diverse bengali-english code-mixed dataset for sentiment analysis. *arXiv e-prints*, arXiv-2408.
- AlHazzani, N., AlDawod, L., AlReshoudi, N., Al-Khalifa, H., & AlDhubayi, L. (2022). Sa7r: A saudi dialect irony dataset. *Proceedings of the 5th Workshop on Open-Source Arabic Corpora and Processing Tools with Shared Tasks on Quran QA and Fine-Grained Hate Speech Detection*, 60–70.
- Baruah, H., Singh, S. R., & Sarmah, P. (2023). Assamese back transliteration-an empirical study over canonical and non-canonical datasets. In *Proceedings of the 37th pacific asia conference on language, information and computation* (pp. 801–808).
- Basri, R., Mridha, M. F., Hamid, M. A., & Monowar, M. M. (2021). A deep learning based sentiment analysis on bang-lish disclosure. *National Computing Colleges Conference (NCCC)*, 1–6.
- Das, R. A., Islam, M., & Haque, Z. (2021). Bangla abusive text detection using machine learning. *Journal of Computational Linguistics*, 15(3), 245–262.
- Duong, H. T., & Nguyen-Thi, T. A. (2021). A review: preprocessing techniques and data augmentation for sentiment analysis. *Computational Social Networks*, 8(1), 1–33.
- Fahim, M., Shifat, F., Haider, F., Barua, D., Sourove, M., Ishmam, M., & Bhuiyan, M. (2024). Banglatlit: A benchmark dataset for back-transliteration of romanized bangla. In *Findings of the association for computational linguistics: Emnlp 2024* (pp. 14656–14672).
- Haider, F., Shifat, F. T., Ishmam, M. F., Barua, D. D., Sourove, M. S. U. R., Fahim, M., & Alam, M. F. (2024). Bant: A multi-label hate speech detection dataset for transliterated bangla. *arXiv preprint arXiv:2410.13281*.

- Hasan, J., Datta, S., & Debnath, A. (2023). Character-level bangla text-to-ipa transcription using transformer architecture with sequence alignment. *arXiv preprint arXiv:2311.03792*.
- Hoque, R. A., & Salma, N. (2023). Detecting depression levels from bengali social media text. *arXiv e-prints*, arXiv-2301.
- Hori, T., Cho, J., & Watanabe, S. (2018). End-to-end speech recognition with word-based rnn language models. In *2018 IEEE Spoken Language Technology Workshop (SLT)* (pp. 389–396).
- Iqbal, M. A., Das, A., Sharif, O., Hoque, M. M., & Sarker, I. H. (2022). Bemoc: A corpus for identifying emotion in bengali texts. *SN Computer Science*, 3(2), 135.
- Islam, M. A., Towhiduzzaman, M., Bhuiyan, M. T. I., Maruf, A. A., & Ovi, J. A. (2022). Banel: An encoder-decoder based bangla neural lemmatizer. *SN Applied Sciences*, 4(5), 138.
- Kasai, J., Sakaguchi, K., Bras, R. L., Radev, D., Choi, Y., & Smith, N. A. (2022). A call for clarity in beam search: How it works and when it stops. *arXiv preprint arXiv:2204.05424*.
- Li, B., Liu, H., Wang, Z., Jiang, Y., Xiao, T., Zhu, J., . . . Li, C. (2020). Does multi-encoder help? a case study on context-aware neural machine translation. *arXiv preprint arXiv:2005.03393*.
- Mahmud, H., Mahmud, H., & Rashid, M. R. A. (2024). Enhancing sentiment analysis in bengali texts: A hybrid approach using lexicon-based algorithm and pretrained language model bangla-bert. *arXiv preprint arXiv:2411.19584*.
- Mondal, S. K., Wang, C., Chen, Y., Cheng, Y., Huang, Y., Dai, H.-N., & Kabir, H. D. (2024). Enhancement of english-bengali machine translation leveraging back-translation. *Applied Sciences*, 14(15), 6848.
- Nanayakkara, R., Nadungodage, T., & Pushpananda, R. (2022). Context aware back-transliteration from english to sinhala. *2022 22nd International Conference on Advances in ICT for Emerging Regions (ICTer)*, 051-056.
- Nath, B., Sarkar, S., Mukhopadhyay, S., & Roy, A. (2024). Improving neural machine translation by integrating transliteration for low-resource english-assamese language. *Natural Language Processing*, 1–22.
- Patwa, P., Sitaram, S., & Subramaniam, A. (2020). Comparative analysis of pre-trained transformer models for sentiment analysis. *Journal of Artificial Intelligence and Applications*, 6(3), 121–138.

- Rahman, M. A., Begum, M., Mahmud, T., Hossain, M. S., & Andersson, K. (2024). Analyzing sentiments in elearning: A comparative study of bangla and romanized bangla text using transformers. *IEEE Access*.
- Raihan, M., Hani Tanmoy, U., Binte Islam, A., North, K., Ranasinghe, T., Anastasopoulos, A., & Zampieri, M. (2023). Offensive language identification in transliterated and code-mixed bangla. *arXiv e-prints*, arXiv-2311.
- Roberts, A., & Raffel, C. (2020). *Exploring transfer learning with t5: The text-to-text transfer transformer*. *google ai blog*. Febr.
- Salaev, U., Kuriyozov, E., & Gomez-Rodriguez, C. (2022). A machine transliteration tool between uzbek alphabets. *arXiv e-prints*, arXiv-2205.
- Shiam, A. A., Redwan, S. M., Kabir, M. H., & Shin, J. (2023). A neural attention-based encoder-decoder approach for english to bangla translation. *Computer Science Journal of Moldova*, 91(1), 70–85.
- Shibli, G. S., Shawon, M. T. R., Nibir, A. H., Miandad, M. Z., & Mandal, N. C. (2023). Automatic back transliteration of romanized bengali (banglish) to bengali. *Iran Journal of Computer Science*, 6(1), 69–80.
- Sultana, B., & Mamun, K. A. (2024). Enhancing bangla-english code-mixed sentiment analysis with cross-lingual word replacement and data augmentation. *arXiv e-prints*, arXiv-2408.
- Talafha, B., Abuammar, A., & Al-Ayyoub, M. (2021). Atar: Attention-based lstm for arabizi transliteration. *International Journal of Electrical and Computer Engineering*, 11(3), 2327.
- Tareq, M., Islam, M. F., Deb, S., Rahman, S., & Al Mahmud, A. (2023). Data-augmentation for bangla-english code-mixed sentiment analysis: Enhancing cross linguistic contextual understanding. *IEEE Access*, 11, 51657–51671.
- Vania, C. (2020). On understanding character-level models for representing morphology.
- Ye, K., Jiang, H., Rostamizadeh, A., Chakrabarti, A., DeSalvo, G., Kagy, J.-F., ... Kumar, S. (2024). Spactor-t5: Pre-training t5 models with span corruption and replaced token detection. *arXiv preprint arXiv:2401.13160*.