CSE499A-SENIOR DESIGN

Comparative Exploration of Preprocessing Techniques on Transformers and Conventional Classifiers for Low-Resource Bengali Language

Shahriyar Zaman Ridoy
2022299642

Jannat Sultana
2022217642

Mohammed Arif Uddin
2012967042

Md Hasibur Rahman
2012558042

Zinnat Fowzia Ria
1931343042

1. Abstract

The low-resource Bengali language poses a number of obstacles for preprocessing techniques on Transformers and traditional classifiers, which are thoroughly compared in this research. Specifically, our results demonstrate that applying 'cleantext' as a stand-alone preprocessing methodology consistently yields better results(almost 2% better accuracy) than other approaches, with increased accuracy. Surprisingly, a significant drop in accuracy is shown when 'cleantext' is combined with popular preparation techniques like lemmatization and stop word removal. For best results in low-resource language contexts, this finding emphasizes the subtle effects of preprocessing processes and emphasizes the significance of carefully choosing and fine-tuning techniques.

2. Introduction

In the modern digital era, textual data is an indispensable asset, comprising a mosaic of information gathered from diverse sources. Its in-depth analysis produces crucial insights that push innovation and research, and it acts as a cornerstone for the quick advances in natural language processing (NLP) that have been witnessed, greatly accelerating the growth of technology [1]. The rapid expansion of the internet has led to a significant increase in the availability of vast amounts of textual data. This abundance of data stems from various sources such as social media, online forums, blogs, news stories, and other platforms. However, this data's sheer volume and unstructured nature present considerable challenges, mainly when cleaning and preparing it for different NLP tasks[2]. The cleaning procedure is critical for enhancing data quality and dependability and guaranteeing proper analysis and interpretation.

In the realm of textual data cleaning, researchers frequently initiate the process using rule-based methodologies encompassing the elimination of stop words, symbols, punctuation marks, emojis, and HTML elements [3]. However, this reliance on automated, rule-based strategies may unintentionally remove essential data from a corpus [4]. Additionally, stemming techniques [5], commonly employed in text cleaning, tend to significantly alter a sentence's emotional context when dissimilar words are unified [6]. These traditional approaches, rooted in predefined rules, often fail to capture the intricate contextual nuances in the data, thereby introducing irrelevant noise and diminishing data relevance. The context within a sentence constitutes the surrounding facts, situations, or conditions that lend meaning and comprehension to the words and phrases utilized [7]. It becomes essential to identify and understand these complex contextual components to customize the cleaning procedure to meet the particular requirements of every dataset. A thorough grasp of these intricate contextual aspects is essential to orchestrate a more precise and relevant data-cleaning method and improve the quality and relevance of the processed data.

Refining data for languages with limited linguistic resources, such as Bengali, remains challenging despite the advancements in contextual models like BERT [8], primarily studied in English. Bengali, the world's seventh most spoken language, offers a rich linguistic

heritage, presenting challenges and opportunities for Natural Language Processing (NLP) [9]. Yet, a crucial gap in low-resource language landscapes like Bengal is the need for context-aware data-cleaning algorithms that respect linguistic nuances. The concern within low-resource languages revolves around conventional data-cleaning methods potentially erasing essential information. The need for more linguistic resources compounds these challenges, making indiscriminate data cleaning a risky proposition. While refining Bengali data is imperative, it demands a delicate balance, preserving crucial linguistic nuances while upholding data quality. The scarcity of linguistic tools for Bengali accentuates the urgency to investigate how data-cleansing practices impact contextual model efficiency and performance in these specific contexts. This study holds significance on two fronts. Firstly, it addresses the critical need for optimized methodologies in preserving and enhancing linguistic diversity through context-aware data cleaning for low-resource languages. Secondly, it underscores the importance of maintaining contextual integrity within sentences, particularly in underrepresented linguistic contexts, enhancing their usability and accessibility.

2.1 Unique Contributions

This research aims to establish a unique context-aware data-cleaning pipeline for the low-resource language Bengali by evaluating various data-cleaning strategies. Our main goal was to remove noise from data by preserving the context of the sentence, which would ultimately increase the accuracy of contextual models. Our major contribution can be summarized as follows.

- In initiating the development of our contextual data cleaning pipeline, our primary step involved meticulously curating a wide array of Bengali datasets. We sought out binary and multiclass datasets, carefully selecting a varied mix. We have used five different Bengali datasets. This thoughtful selection process was crucial to facilitate thorough and equitable comparisons in our subsequent evaluations and analyses.
- We've conducted extensive training using both contextual and non-contextual models within the realms of machine learning, deep learning, and transformer-based learning approaches. This training assessed our dataset's performance in its raw, uncleaned state. Currently, the most accurate model is a contextual transformer-based model, a BERT-based model, which is considered the state of the art. Our comparative analysis highlights how each dataset fares when contextual and non-contextual models are utilized, offering a clear insight into their performance differences.

- This research focuses on assessing traditional textual data-cleaning methods. We've implemented four traditional data-cleaning approaches and examined their impact on individual sentences. These approaches were then employed to train both contextual and non-contextual models. Our analysis involves comparing how each cleaning method influences the accuracy of these models, shedding light on their respective contributions to either enhancing or diminishing model performance.
- In this research, we've integrated context-aware data-cleaning techniques. We've devised six strategies aimed at minimizing sentence noise while preserving context. Our focus was to demonstrate the significance of each method in maintaining sentence context while diminishing text noise. These strategies were applied to our chosen datasets, allowing us to assess their impact on enhancing contextual and non-contextual models. Through comparative analysis, we've evaluated how each strategy contributes to improving our contextual models' performance while highlighting their effects on non-contextual models.
- We've developed a context-aware data pipeline by tailoring six context-aware data-cleaning strategies designed to effectively clean datasets while preserving their contextual integrity, particularly in low-resource language setups. Moreover, we've outlined the algorithms for context-aware data cleaning, making them accessible for implementation by anyone looking to employ these techniques.

In summary, our context-aware data-cleaning pipeline is essential because it ensures the preservation of vital contextual information while preparing data for NLP tasks. This approach significantly contributes to NLP models' accuracy, generalizability, and inclusivity, especially in diverse linguistic contexts and resource-constrained environments.

2.2 Research Gap

The research backdrop for low-resource languages often lacks robust methodologies personalised to preserve contextual integrity during data cleaning. In order to fill this gap, this work introduces a context-aware data-cleaning pipeline that goes above and beyond traditional cleaning procedures, advancing Bangla NLP approaches. Through a performance comparison with conventional cleaning methods, the study advances our knowledge of how well context-aware approaches refine data for higher model accuracy.

3. Related Work

With the development of sophisticated contextual models, text data cleaning has become an increasingly important aspect of natural language processing. In this section, we undertake a comprehensive review of pertinent studies, delving into diverse text-cleaning techniques, their impact on model accuracy, and the intricate task of preserving contextual nuances.

To explore the impact of diverse text-cleaning methods, Zaki et al. [10] undertake an investigation employing fourteen text-cleaning techniques applied to datasets sourced from Twitter, Facebook, and YouTube. The authors systematically use these techniques in a specified sequence. Utilizing Machine Learning Models like SVM, the study evaluates the fluctuation in accuracy for sentiment classification with the proposed preprocessing strategies. The findings indicate that employing all preprocessing approaches consistently leads to achieving an accuracy of 82.57% using unigram representations. Despite the demonstrated effectiveness of the proposed preprocessing strategy on the chosen dataset, a thorough exploration employing how each cleaning technique is affecting the text, specifically in a preserving context, needs to be included. Also, investigation using contextual models like BERT is absent in this study. In a related study, Singh et al. [11] investigated different text-cleaning strategies for sentiment classification tasks in Twitter datasets. The authors examine various combinations of proposed text cleaning techniques and observe that incorporating URL feature preservation, repeated letters normalization, and negation transformation enhances sentiment classification accuracy. Conversely, the accuracy diminishes when employing stemming and lemmatization. Additionally, augmenting the initial feature space with bigrams and emotional features yields superior outcomes. Although Singh et al. show the impact of different text-cleaning techniques, their research needs an in-depth analysis of why these techniques help these models gain or lose accuracy. Also, they should have discussed the context of the sentence or the performance of the contextual model while using these techniques.

To show the importance of cleaning text for modern contextual models, Kumar et al.[12] conducted research using comparative analysis of how noisy text significantly degrades the performance of BERT. They chose some benchmark datasets and created artificial noise in those datasets, such as spelling mistakes and typos. Their result shows that BERT's performance on fundamental NLP tasks like sentiment analysis and textual similarity drops significantly in the presence of noise on benchmark datasets. They further revealed that the reason for the performance drop is how BERT's tokenizer (WordPiece) handles the misspelled words. However, their paper emphasized the importance of cleaning text but did not discuss how to clean the text and whether cleaning the text to preserve context is essential. Another research on cleaning text based on spelling mistakes and typos was

conducted by Sun et al. [13]. They proposed a contextual text denoising algorithm based on the ready-to-use masked language model. Their algorithms use the power of contextual models like BERT, which can recover the noisy text from the contextual information without any training or data. They have demonstrated that using their denoising algorithms with masked language models can remove noise from the sentences and improve the accuracy of various NLP tasks. However, the denoising process can be computationally expensive and time-consuming depending on the algorithm's complexity and the text data's size. For a more robust context-aware data cleaning, Khan et al. [14] proposed a context-aware text normalization technique. They used Textual Variations Handler (TVH) to address word enlargement, abbreviations, word shortening, spelling errors, phonetic substitution, dialectal usage, and word deletion problems of social media text in a context-aware manner. This research aims to introduce a hybrid normalization technique that effectively ensures that information obtained from noisy text data can be utilized in the desired form. This study integrates the TVH application with a deep-learning state-of-the-art (SOTA) based text analysis method to improve its performance in analyzing noisy SM text data. The simulation results show that the proposed scheme is promising in terms of precision, recall, accuracy, and F1 scores in the analysis of informal texts on social media. Their research shows no comparison with other text normalization methods that have been used in previous studies. This study only compares the performance of the proposed method with the same procedure without normalization. Although they said that their proposed methods are context-aware, they didn't provide any example of how their text normalization technique is preserving the context of the sentence.

A recent study by Siino et al. [15] examined the relevance of text preprocessing by comparing its impact on Text Classification performance across contextual and non-contextual models. They have explored a total of 15 various text cleaning techniques, applied them to publicly available datasets, and evaluated nine machine-learning models, including modern Transformers. Their results indicate that an informed choice of preprocessing strategy significantly influences classification accuracy, with improvements of up to 25% observed, especially evident in machine learning models like Naïve Bayes. Transformers and traditional models display substantial sensitivity to text preprocessing, emphasizing its continued relevance in enhancing Text Classification performance. Their works show an in-depth analysis of the effects of different text cleaning strategies for contextual and noncontextual models and suggest that without any cleaning, Transformer models can perform at an optimal level. However, their study lacks information on how each of these different text-cleaning strategies affects the context of the sentence and how they help both contextual and non-contextual models in terms of accuracy.

Despite the wealth of data-cleaning approaches discussed, a notable gap exists in the realm of context-aware data-cleaning techniques, especially for low-resource languages like Bengali. While a few context-aware methods for English are available, their efficacy in preserving context remains a subject of exploration. Moreover, the scarcity of research on data cleaning for low-resource languages underscores the need for further investigation in this domain.

This expanded overview aims to provide a nuanced exploration of studies in text data cleaning, setting the stage for the proposed context-aware text data cleaning pipeline discussed later in this paper.

4. Methodology

In this experiment, we developed a context-aware data-cleaning pipeline for low-resource language that uses multiple cleaning and transformation techniques to improve sentence quality while maintaining contextual integrity. We integrated four distinct noise removal methods to construct this pipeline. We used it to train both context and non-context-based models, and then we evaluated their performance against results from traditional data-cleaning techniques. We aimed to assess our recently created context-aware pipeline against existing methods regarding how well it refined data and increased the accuracy of context-based models.

Our experiment was structured into distinct phases, as shown in Figure 1. Initially, we trained our dataset using context and non-context-based models without cleaning procedures. Subsequently, we applied traditional cleaning techniques to refine the data and introduced these cleaned sets separately on context and non-context-based models. Finally, we implemented our context-aware data-cleaning pipeline to refine the data, then trained the context-enhanced cleaned dataset on both models.

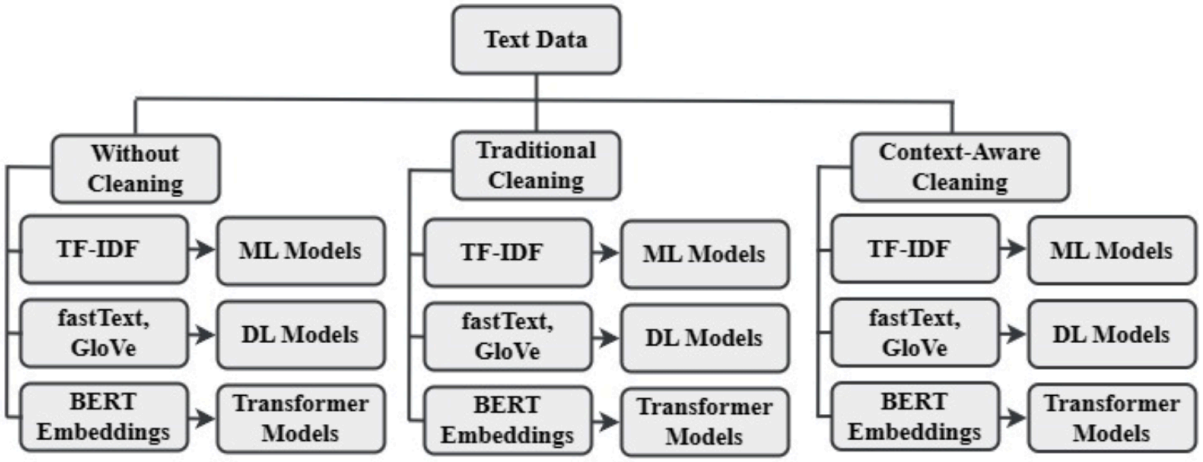

Figure 1: This figure provides a visual representation of the different types of models used (Non-Context-Based, Context-Based, and Deep Learning), along with the techniques employed within each category.

4.1 Dataset

We've curated a diverse set of established datasets specifically designed for the low-resource Bengali language. Our meticulous selection process prioritized including binary and multiclass datasets, ensuring a comprehensive and balanced comparison in our evaluations. In the following discussion, we'll delve into the specifics of our chosen datasets to provide a clear overview of our working scope.

- **BEmoC:** Anger, fear, surprise, sorrow, joy, and disgust are among the six main emotion categories labeled on 7000 Bengali texts that make up BEmoC, a corpus for recognizing emotion in Bengali literature [16]. Pre-processing, labeling, verification, and data crawling are the four main processes in the corpus generation process.

- **SentNoB**: This dataset comprises public comments on news and videos collected from social media covering 13 different domains, including politics, education, and agriculture. These comments are labeled with polarity labels: positive, negative, and neutral [17].

- **UBMEC:** Based on user reviews, the Unified Bangla Multi-class Emotion Corpus (UBMEC) has been created for Bangla. It combines two publicly accessible Bangla emotion corpora previously released, BNEmo and BEmoC, with an extra manually tagged corpus we created to provide a sufficiently developed Bangla emotion corpus

[18]. It is compiled from various industries, including software, cuisine, sports, politics, and entertainment. This corpus is divided into six unified classes: anger, disgust, fear, joy, sadness, and surprise.

- **EmoNoBa**: This dataset comprises 22,698 manually annotated Bangla public comments extracted from various social media platforms [19]. It spans 12 distinct domains, including Personal, Politics, and Health. The words are meticulously labeled, classifying emotions into six fine-grained categories aligned with the Junto Emotion Wheel.

- **BanFakeNews**: This dataset comprises approximately 50,000 annotated news articles categorized into binary classes [20]. These articles were gathered from various online news sources in Bangladesh, offering a comprehensive collection for analysis and classification tasks.

4.2 Word Representations

In natural language processing (NLP), word representations are fundamental in understanding and processing language. These representations aim to encode words in a manner that captures their semantic, syntactic, and contextual meanings within a given context or corpus. Various techniques and models have been developed to achieve effective word representations:

- **TF-IDF**: A statistical metric called TF-IDF [21] is used in information retrieval to determine how important a word or phrase is inside a document in relation to a group of documents. It gauges the importance of a phrase by considering how frequently it appears in a particular text and how seldom it occurs throughout the corpus.
  For instance, if a term frequently appears within a document but rarely occurs in other documents, TF-IDF considers it significant for that particular document. Conversely, common words across the corpus receive lower TF-IDF scores as they lack distinctive contextual relevance.

- **GloVe**: GloVe [22] is a method that generates word embeddings by leveraging the co-occurrence statistics of words within a corpus. It constructs word vectors based on the distributional patterns of word co-occurrences, aiming to capture the semantic

relationships between words. GloVe represents words in a vector space where distances between word vectors reflect their semantic similarities or associations based on their co-occurrence frequencies.

- **fastText**: fastText [23], an approach for word representation, operates by considering subword information such as character n-grams. It encodes linguistic structures and representations by breaking words into smaller subword units. This technique enables fastText to handle out-of-vocabulary words by constructing their representations from known character sequences, facilitating a better understanding of complex morphologies and languages.

- **BERT:** BERT [24], a transformer-based model, captures contextual word embeddings by considering the bidirectional context of words within a sequence. It employs a deep bidirectional architecture, allowing it to interpret words based on their relationships with surrounding words in both directions. This bidirectional understanding helps BERT generate rich, contextually aware representations, making it highly effective for various natural language understanding tasks.

4.3 Baseline Model Training

We've employed context-based and non-context-based models to train our datasets without cleaning. We utilized deep learning models like CNN, LSTM, and BiLSTM alongside traditional machine learning models such as SVM, Random Forest, XGBoost, and Naive Bayes for the non-context-based models. To represent words in these models, we relied on TF-IDF for traditional machine learning models and fastText and GloVe embeddings for deep learning models, which operate independently of contextual information. For the context-based models, we have incorporated BanglaBERT, Multilingual BERT, IndicBert, and XLM-RoBERTa, which rely on BERT embeddings to grasp the contextual information within sentences.

- **SVM:** SVM [25] is a supervised learning model for classification and regression tasks. They work by finding the optimal hyperplane that separates data into different classes. SVMs are adept at handling text classification tasks, such as sentiment analysis or document categorization, by effectively separating textual features into

other classes. They excel in scenarios where the text data might not be linearly separable and require complex decision boundaries.

- **Random Forest:** Random Forest [26] is an ensemble learning method with multiple decision trees. Each tree in the forest operates independently and contributes to the final prediction. Random Forest can be employed in text analysis for tasks like text classification. It processes textual features efficiently, making it useful when dealing with a high-dimensional feature space common in text-based datasets.

- **XGBoost:** XGBoost [27] is a gradient-boosting algorithm that builds multiple decision trees sequentially to improve predictive performance. It's commonly used in tasks like topic modeling, text classification, or text summarization. XGBoost efficiently handles missing data and performs feature selection, making it suitable for text-based datasets.

- **Naive Bayes:** Naive Bayes [28] is a probabilistic classifier based on Bayes' theorem, assuming independence between features. It calculates the probability of a label given input features. In text analysis, Naive Bayes is popular for text classification tasks like spam detection, sentiment analysis, or document categorization. It's known for its simplicity and efficiency in processing large volumes of text data.

- **CNN:** CNNs [29] are deep neural networks primarily used in image recognition but are also effective in text analysis. They use convolutional layers to extract features from input data. In NLP, CNNs can automatically learn relevant features from text, making them suitable for tasks like text classification, sentiment analysis, or document categorization.

- **LSTM:** LSTMs [30] are a type of recurrent neural network designed to overcome the limitations of traditional RNNs in capturing long-term dependencies in sequential data. In text analysis, LSTMs are used for tasks like language modeling, text generation, or sentiment analysis. They retain information over longer sequences, making them suitable for analyzing text data with complex structures.

- **BiLSTM:** By processing input sequences in both forward and backward directions, Bidirectional Long Short-Term Memory (Bi-LSTM) [31] networks improve upon the capabilities of conventional LSTMs. Using a bidirectional strategy, the network can simultaneously gather contextual data from previous and upcoming time steps.

Bi-LSTMs are especially good for named entity recognition and sentiment analysis because they combine the advantages of bidirectional processing and long-range dependency capture from LSTMs. This allows Bi-LSTMs to comprehend complex patterns and relationships within sequential data.

- **BanglaBERT:** In natural language processing, Bengali is a language that faces limitations in available resources. BanglaBERT, a BERT-based NLU model, was trained on a 27.5 GB dataset from 110 prominent Bengali websites. It leverages the ELECTRA model and utilizes the Replaced Token Detection (RTD) method. Fine-tuned models using BanglaBERT have demonstrated superior performance across various NLP tasks tailored explicitly for Bengali language applications.

- **Multilingual BERT:** Expanding the capabilities of the original BERT architecture to several languages, Google developed Multilingual BERT (mBERT) [32], a crucial model for natural language processing. Contextualized representations of words and phrases in many languages are produced by mBERT, which was trained on a large corpus of material from various linguistic origins. To facilitate cross-lingual transfer learning, mBERT's multilingual pre-training approach enables it to establish a shared representation space.

- **IndicBert:** The multilingual ALBERT [33] model known as IndicBERT [34] covers twelve main Indian languages, including Bengali. It is pre-trained on a unique corpus of around 9 billion tokens and assessed on various tasks. Indic-bert delivers a performance that is on par with or better than other well-known, publicly accessible multilingual models but has around 10 times fewer parameters.

- **XLM-RoBERTa:** Cross-lingual Language Model - RoBERTa, or XLM-RoBERTa [35], is a robust transformer-based model for multilingual NLP applications. It can capture language-agnostic contextual representations because it was trained on a large multilingual dataset. It understands and generates text across different languages by learning from a diverse, multilingual dataset. The model is well suited for cross-lingual transfer learning since it performs exceptionally well at comprehending and producing representations for text in multiple languages.

4.4 Traditional Cleaning and Training

We have selected the common approach to clean text data for traditional data cleaning. These techniques are more likely rule-based as they don't consider the context of the sentence while cleaning.

- **Eliminating Symbols and Punctuation:** This technique aims to enhance the cleanliness of text data by removing symbols, punctuation marks, and non-alphanumeric characters. The goal is to refine the text by eliminating distracting elements. However, this method risks losing crucial contextual cues within sentences, potentially altering the original content's meaning or emphasis. For instance, removing punctuation can affect sentence structure or tone interpretation.

Original Text	Cleaned Text
Bengali: বৃষ্টি পড়ছে! এই সময়ে গরম চা খুব ভালো লাগে। 😊 **English**: It's raining! Hot tea is very nice at this time.😊	**Bengali**: বৃষ্টি পড়ছে এই সময়ে গরম চা খুব ভালো লাগে **English**: Hot tea is very good when it is raining

- **Removing Stop Words:** Stop words, commonly used words like "the," "and," "is," etc., are often removed to streamline datasets and improve computational efficiency. While eliminating stop words can focus the analysis on more meaningful terms, it also runs the risk of discarding contextually significant words. This removal might alter the text's intended meaning and affect tasks like sentiment analysis or context-based understanding.

Original Text	Cleaned Text
Bengali: তুমি যা বললে তা সত্যি। **English**: What you said is true.	**Bengali**: বললে সত্যি। **English**: It's true.

- **Stemming:** Stemming [36] involves reducing words to their base or root form, aiming to standardize word variations by removing prefixes or suffixes. This process consolidates different forms of the same word into a single representation, aiding analysis. However, stemming might not accurately capture the intended meaning as it can produce word stems that aren't valid or may not reflect the context properly.

Original Text	Cleaned Text

Bengali: তোমার ছবিগুলি খুব সুন্দর। **English:**Your pictures are very beautiful.)	**Bengali**: তোমার ছবি সুন্দর। **English:** Your picture is beautiful.

- **Removing HTML Tags or URLs:** Eliminating HTML tags or URLs improves readability and ensures a cleaner dataset. However, this approach might inadvertently remove valuable information essential for web-based analyses. For instance, removing URLs can eliminate crucial source or citation information, impacting the comprehensiveness of the dataset for tasks like web content analysis or link-based studies.

Original Text	Cleaned Text
Bengali: নতুন রেসিপি খুঁজছি? এখানে দেখুন: www.example.com/recipes **English:** Looking for new recipes? See here: www.example.com/recipes)	Bengali: নতুন রেসিপি খুঁজছি? এখানে দেখুন: **English:** Looking for new recipes? See here:)

These conventional techniques play a part in removing noise from the data. Still, they risk compromising language's nuanced context and subtleties, potentially impacting the accuracy and depth of subsequent analyses. We have trained our context-based and non-context-based models using these techniques, which we discussed in section 5.3.

4.5 Context-Aware Data Cleaning and Training

Unlike traditional rule-based methods, which often affect sentence context, we've adopted context-preserving techniques for data cleaning. These methods aim to maintain the original context of sentences while minimizing noise in the dataset.

- **Using Tags for Symbols and Punctuation:** This method involves tagging symbols and punctuation inside the text to preserve their meaning and value for sentence structure instead of removing them completely. We are using tags like <PUNC> for punctuation, <DIGIT> for digits, and <NUMBER> for numbers. The original context of sentences is preserved by using these tags to hold onto the aspects of facilitating a more thorough comprehension during sentiment analysis.

Original Text	Cleaned Text

Bengali: বৃষ্টি পড়ছে! এই সময়ে গরম চা খুব ভালো লাগে। **English:** It's raining! Hot tea is very nice at this time.)	Bengali: বৃষ্টি পড়ছে <PUNC> এই সময়ে গরম চা খুব ভালো লাগে <PUNC> **English:** It's raining <PUNC> Hot tea is so good at this time <PUNC>)

- **Using Tags for HTML and URLs:** Rather than removing HTML tags or URLs, tagging these elements helps retain their relevance within the text. This preservation ensures that information embedded in HTML tags or URLs is not lost, which can be vital, especially in web-based analyses. We have used <URL> and <EMAIL> tags to replace them with the original HTML tags or URLs.

Original Text	Cleaned Text
Bengali: নতুন রেসিপি খুঁজছি? এখানে দেখুন: www.example.com/recipes **English:** Looking for new recipes? See here: www.example.com/recipes)	**Bengali**: বৃষ্টি পড়ছে <PUNC> এই সময়ে গরম চা খুব ভালো লাগে <PUNC> **English:** It's raining <PUNC> Hot tea is so good at this time <PUNC>)

- **Preserving Emoji:** Emojis, carrying contextual cues and emotions, are retained within the text. Preserving emojis ensures they contribute to the overall context without losing their intended meanings, which can be significant for sentiment analysis or conveying emotions in text.

Original Text	Cleaned Text
Bengali: আমি ভাল আছি	🙂 **English**: I am fine. 🙂	**Bengali**: আমি ভাল আছি	🙂 **English**: I am fine. 🙂

- **Removing Less Important Words:** Instead of simply removing common words (stop words), our approach involves using TF-IDF [3] to assess word importance based on their occurrence in documents and the entire dataset. TF-IDF helps identify and eliminate less contextually significant words, streamlining the dataset by selectively removing less crucial terms while safeguarding essential contextual information.

Original Text	Cleaned Text
Bengali: 1. এই ফোনের ক্যামেরা মান অসাধারণ, তবে ব্যাটারি লাইফ সন্তোষজনক নয়। 2. আমি এই ফোনের ডিজাইনটি ভালোবাসি। এটি প্রিমিয়াম লাগে এবং সুন্দর দেখায়। 3. আমি এই ফোনের ডিজাইনটি ভালোবাসি। এটি প্রিমিয়াম লাগে এবং সুন্দর দেখায়। **English:**	**Bengali:** 1. ক্যামেরা মান অসাধারণ, তবে ব্যাটারি লাইফ সন্তোষজনক নয়। 2. ব্যাটারি লাইফ অবিশ্বাস্য, কিন্তু ক্যামেরাটি ভালো হতে পারে। 3. আমি ডিজাইনটি ভালোবাসি। এটি প্রিমিয়াম লাগে এবং সুন্দর দেখায়। **English:** 1. Camera quality is great, but battery

1. The camera quality of this phone is great, but the battery life is not satisfactory. 2. I love the design of this phone. It feels premium and looks good. 3. I love the design of this phone. It feels premium and looks good.	life is not. 2. Battery life is incredible, but the camera could be better. 3. I love the design. It feels premium and looks good.

- **Spelling Correction:** This step involves spell-checking to ensure the accuracy of words within the text. Correcting spelling errors helps maintain the integrity of the language and prevents misinterpretation due to misspelled words, thus preserving the intended context.

Original Text	Cleaned Text
Bengali: আমি শিক্ষিত হতে চাই, আমি কমল ধরতে চাই। **English:** I want to be educated, I want to hold the lotus.)	**Bengali**: আমি শিক্ষিত হতে চাই, আমি কলম ধরতে চাই। **English:** I want to be educated, I want to hold the pen.

- **Word Translation:** In multilingual datasets, ensuring contextual accuracy involves accurately translating words to maintain meaning across languages. In Bengali datasets, we identified English words written in Bengali script and translated them into Bengali. This meticulous process aims to maintain the original context as closely as possible.

Original Text	Cleaned Text
Bengali: আমি তোমাকে ভাল Friend ভাবি **English:** I consider you a good friend	**Bengali:** আমি তোমাকে ভাল বন্ধু ভাবি **English:** I consider you a good friend

We have applied the above techniques to clean our datasets contextually. After doing the cleaning, we have applied the models we have discussed in section 5.2. We have applied context-based and non-context-based models to visualize how context-aware data cleaning methods perform compared to traditional cleaning methods in both context- and non-context-based models.

4.6 Context-Aware Data Cleaning Pipeline

We've established a comprehensive data pipeline encompassing various context-aware data-cleaning strategies outlined in section 5.4. Figure 1 illustrates the step-by-step process wherein datasets undergo systematic cleaning while preserving context before being utilized for training context-based models. Initially, our unclean text undergoes the **replaceSymbolsAndPunctuation** step, replacing symbols and punctuations with tags. Following this, the **preserveHTMLAndURLs** function replaces URLs and HTML elements with tags while ensuring the integrity of the text's context. **PreserveEmojis** aims to retain images represented by emojis alongside the text. Subsequently, the **removeLessImportantWords** function utilizes TF-IDF to remove less crucial words. Further refinement involves **spellCorrection** and **wordTranslation** steps, correcting spelling errors, and translating non-Bengali words. Finally, the thoroughly cleaned text is channeled into context-based models such as BERT for classification.

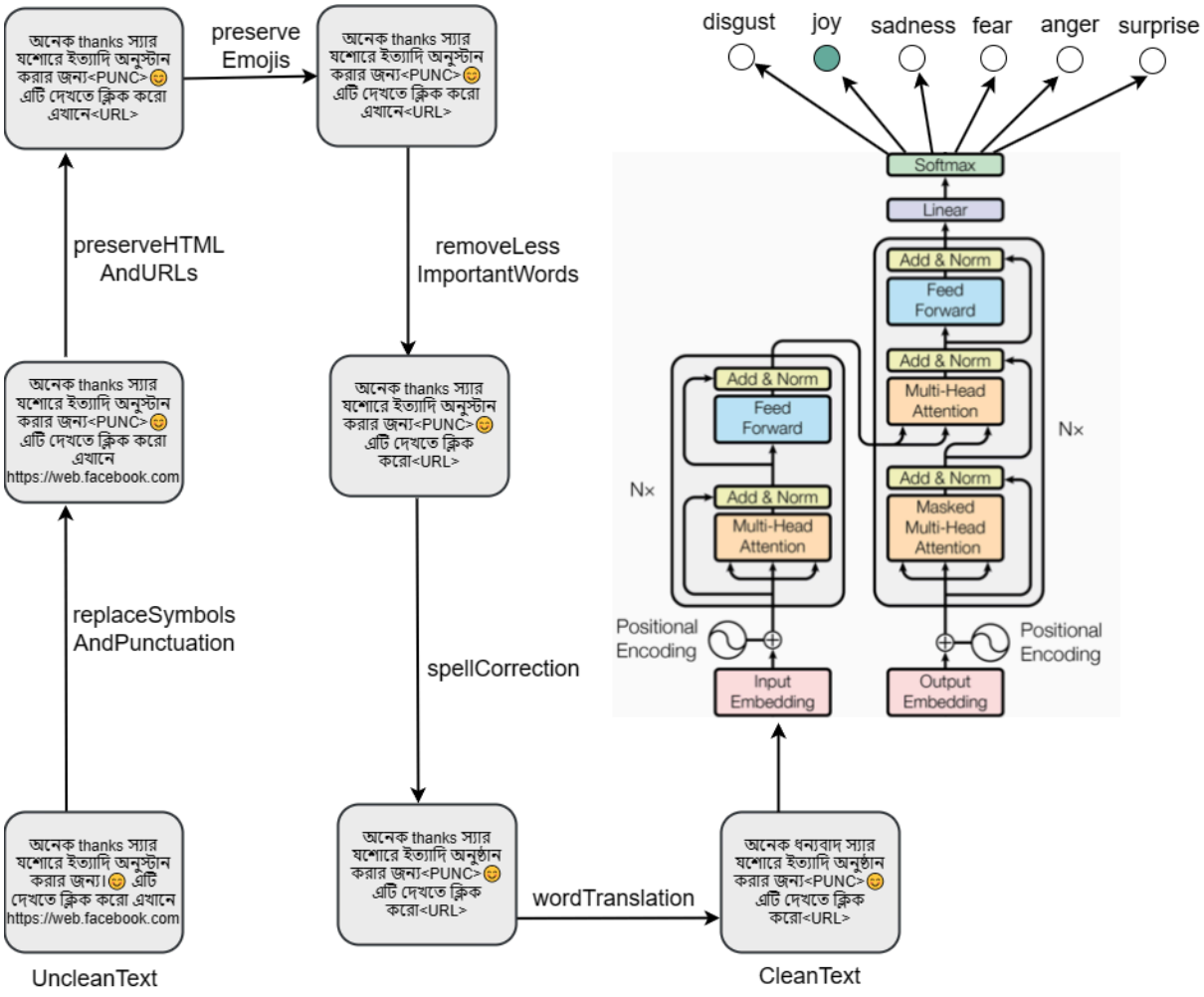

Figure 2: This figure illustrates how data is cleaned in each step by preserving the content of the data and finally feeding the clean data to context-based model training

We have also developed a context-aware data-cleaning algorithm called CADClearning, through which data will be cleaned following the pipeline illustrated in Figure 1. This algorithm will help clean the data without any prior knowledge of data cleaning.

Algorithm 1 Pseudocode for Context-Aware Data Cleaning Pipeline

```
function CADCleaning(text_data, language_tags)
    cleaned_text_data ← []
    for text in text_data do
        cleaned_text ← replaceSymbolsAndPunctuation(text)
        cleaned_text ← preserveHTMLAndURLs(cleaned_text, language_tags)
        cleaned_text ← preserveEmojis(cleaned_text)
        cleaned_text ← removeLessImportantWords(cleaned_text)
        cleaned_text ← spellCorrection(cleaned_text)
        cleaned_text ← wordTranslation(cleaned_text, language_tags)
        Append cleaned_text to cleaned_text_data
    end for
    return cleaned_text_data
end function
```

Algorithm 1 illustrates the CADCleaning function, which serves to clean text data while aiming to retain the original context of the sentences. It uses a list of text entries for cleaning and language-specific tags to aid context preservation. The function initializes an empty list called **cleaned_text_data** to store the processed text entries. It then proceeds through a loop, iterating over each text entry within the text_data list. Inside the loop, various cleaning operations are performed sequentially on each text entry. These operations involve replacing symbols and punctuation, preserving HTML tags and URLs, retaining emojis, removing less contextually significant words, correcting spelling errors, and translating words based on language tags. Each step modifies the text, preserving essential context while enhancing data cleanliness. After processing each text entry, the cleaned version is appended to the **cleaned_text_data** list. Once all text entries are processed, the function returns the **cleaned_text_data** list, containing the cleaned and contextually preserved text entries, ready for model training.

4.7 Model Training and Evaluation

We've implemented diverse methodologies to assess datasets thoroughly. Our focus spans from innovative context-aware data-cleaning techniques to both traditional and no-cleaning approaches. Our strategy includes an early stopping callback [37] to prevent overfitting and improve generalization by automatically identifying the best training epochs. This automated process significantly streamlines model development, eliminating the need for manual tuning. We've also incorporated cyclic learning rate techniques [38], enhancing model training by dynamically adjusting learning rates within specified bounds.

Additionally, for proper dataset evaluation, we've employed the ktrain library to assess model performance using precision, recall, F1-score, and support metrics for individual classes. This comprehensive evaluation encompasses accuracy, weighted average, and macro average, enabling a thorough understanding of the model's capabilities across various metrics.

Accuracy: It measures the ratio of correctly predicted instances (for example, correctly classified text samples) to the total number of cases in the validation set. It offers an overall assessment of how accurately the model predicts the classes of text samples. However, accuracy might not be the best metric for evaluation, especially when dealing with imbalanced datasets where one class heavily outweighs the others.

$$Accuracy = (True\ Positives + True\ Negatives) / Total\ instances$$

Precision: In text classification, precision represents the model's ability to identify relevant instances within the predicted positive class (e.g., correctly identifying relevant text samples among all predicted positives).

$$Precision = True\ Positives / (True\ Positives + False\ Positives)$$

Recall: It gauges how well the model captures all the relevant instances within a specific class. For text classification, recall signifies the model's ability to find all the relevant text samples belonging to a particular category among all relevant samples.

$$Recall = True\ Positives / (True\ Positives + False\ Negatives)$$

F1-Score: The F1 score balances precision and recall, offering a single metric to assess a model's performance in text classification tasks. It provides an overall evaluation of the model's ability to classify text samples while correctly considering precision and recall.

$$F1\text{-}score = 2 * Precision * Recall / (Precision + Recall)$$

These performance indicators - accuracy, recall, precision, and F1-score - are commonly used to evaluate dataset and their performance on the models. To measure how good or bad our context-aware data-cleaning pipeline performs, we have used these metrics and compared them. This gave us a clear indication of the performance of the context-aware data-cleaning pipeline.

5. Results and Analysis

5.1 Machine Learning

Dataset	Models	CleanText (Removing symbols, punctuations, html tags and urls)	Stop word removal	Lemmatization	Accuracy	F1 Score
sentNob	Random Forest Tree	×	×	×	0.6364	0.5907
		✓	×	×	0.6357	0.5839
		✓	✓	×	0.5883	0.5482
		✓	×	✓	0.6383	0.5853
		✓	✓	✓	0.6095	0.5700
	XGBOOST	×	×	×	0.5928	0.5233
		✓	×	×	0.5864	0.5116
		✓	✓	×	0.5448	0.4771
		✓	×	✓	0.5813	0.5147
		✓	✓	✓	0.5544	0.4844
	NB	×	×	×	0.5813	0.4693
		✓	×	×	0.5781	0.4658
		✓	✓	×	0.5608	0.4517
		✓	×	✓	0.5832	0.4731
		✓	✓	✓	0.5659	0.4571
	SVM	×	×	×	0.6312	0.5789
		✓	×	×	0.6274	0.5732
		✓	✓	×	0.6031	0.5421
		✓	×	✓	0.6351	0.5760
UBMEC	Random Forest Tree	×	×	×	0.4332	0.4065
		✓	×	×	0.4332	0.4057
		✓	✓	×	0.4112	0.3897
		✓	×	✓	0.4234	0.3959
		✓	✓	✓	0.3999	0.3857
	XGBOOST	×	×	×	0.4257	0.4015
		✓	×	×	0.4245	0.3967
		✓	✓	×	0.4056	0.3872
		✓	×	✓	0.4272	0.3983
		✓	✓	✓	0.4101	0.3895
	NB	×	×	×	0.4050	0.3153
		✓	×	×	0.3983	0.3072
		✓	✓	×	0.3920	0.3136
		✓	×	✓	0.3968	0.3067
		✓	✓	✓	0.3947	0.3161
	SVM	×	×	×	0.4516	0.4200
		✓	×	×	0.4474	0.4179
		✓	✓	×	0.4278	0.4022
		✓	×	✓	0.4399	0.4118
	RANDOM FOREST TREE	×	×	×	0.8661	0.8194
		✓	×	×	**0.8675**	**0.8205**
		✓	✓	×	0.8494	0.7991
		✓	×	✓	0.8603	0.8112
		✓	✓	✓	**0.8454**	**0.7847**
	XGBOOST	×	×	×	0.8576	0.8132
		✓	×	×	**0.8607**	**0.8174**
		✓	✓	×	0.8431	0.7770

YouTube Review		✓	×	✓	0.8580	0.8095
		✓	✓	✓	**0.8409**	**0.7717**
	NB	×	×	×	0.8306	0.7399
		✓	×	×	0.8302	0.7368
		✓	✓	×	**0.8333**	**0.7444**
		✓	×	✓	**0.8284**	**0.7318**
		✓	✓	✓	0.8288	0.7353
	SVM	×	×	×	0.8738	0.8317
		✓	×	×	**0.8738**	**0.8317**
		✓	✓	×	0.8557	0.8063
		✓	×	✓	0.8684	0.8234
		✓	✓	✓	**0.8539**	**0.7946**

5.1.1 SentNoB

This study investigates the influence of preprocessing techniques on the performance of four machine learning models—Random Forest Tree, XGBoost, Naive Bayes (NB), and Support Vector Machine (SVM)—using the SentNob dataset. The considered preprocessing techniques include CleanText, Stop Word Removal, and Lemmatization, with various combinations explored for each model.

For Random Forest Tree, the model achieved its accuracy (63.64%) and highest F1-score (59.07%) without preprocessing. However, applying text cleaning resulted in a slight decline in both accuracy (63.57%) and F1-score (58.39%). Combining text cleaning with stop word removal saw a significant decrease to an accuracy of 58.83% and an F1-score of 54.82. Text cleaning with lemmatization demonstrated an increase in highest accuracy (63.83%) and F1-score (58.53). Surprisingly, combining all three techniques led to a notable drop in accuracy (60.95%) and F1-score (57%). These findings underscore the nuanced impact of preprocessing on the Random Forest Tree model's performance, suggesting that the combination of text cleaning and lemmatization proves to be the most effective for this particular task on the SentNob dataset.

For the XGBoost model, the analysis revealed that without preprocessing, the highest accuracy was achieved at 59.28%, accompanied by an F1-score of 52.33%. The introduction of text cleaning resulted in a slight reduction in accuracy to 58.64%, with a corresponding F1-score of 51.16%. Further, when text cleaning was combined with stop word removal, a more pronounced decrease in performance was observed, yielding an accuracy of 54.48% and an F1-score of 47.71%. Text cleaning combined with lemmatization showed a moderate improvement, reaching an accuracy of 58.13% and an F1-score of 51.47%. Surprisingly, the combination of text cleaning, stop word removal, and lemmatization led to a further decline in accuracy to 55.44%, coupled with an F1-score of 48.44%. These findings emphasize the sensitivity of the XGBoost model to different

preprocessing techniques, highlighting the importance of selecting appropriate strategies for optimal performance on the SentNob dataset.

In the case of Naive Bayes, without preprocessing, the model achieved an accuracy of 58.13% and an F1-score of 46.93%. The application of text cleaning resulted in an accuracy of 57.81% and an F1-score of 46.58%, while text cleaning combined with stop word removal led to an accuracy of 56.08% and an F1-score of 45.17%. Text cleaning combined with lemmatization yielded an accuracy of 58.32% and an F1-score of 47.31, and the combination of text cleaning, stop word removal, and lemmatization resulted in an accuracy of 56.59% and an F1-score of 45.71%.

Finally, for Support Vector Machine (SVM), without preprocessing, the model achieved the accuracy of 63.12% and an F1-score of 57.89%. The application of text cleaning resulted in an accuracy of 62.74% and an F1-score of 57.32%, while text cleaning combined with stop word removal led to an accuracy of 60.31% and an F1-score of 54.21%. Text cleaning combined with lemmatization yielded an accuracy of 63.51% and an F1-score of 57.60, and the combination of text cleaning, stop word removal, and lemmatization resulted in an accuracy of 61.72% and an F1-score of 55.68%. These findings emphasize the nuanced impact of preprocessing on SVM performance, indicating that text cleaning with lemmatization proves to be the most effective preprocessing combination on the SentNob dataset.

5.1.2 UMBEC

Model Performance:

Random Forest Tree: The best performance in terms of accuracy of 43.32% and F1 score of 40.65% was attained when employing lemmatization and without stop word removal. Adding clean text, which excludes HTML tags, URLs, punctuation, and symbols, did not appreciably alter the outcomes.

XGBoost: With an accuracy of 42.57% and an F1 score of 40.15%, lemmatization without stop word removal produced the best results, much like Random Forest Tree. CleanText alone did not improve the results.

Naive Bayes (NB): With an accuracy of 40.50% and an F1 score of 31.53%, the best results were obtained utilizing cleanText solely, without any preprocessing. It's interesting to see that lemmatization with no stop word removal produced the second-best results.

Support Vector Machine (SVM): The best result was observed while utilizing lemmatization without stop word removal, with an accuracy of 45.16% and an F1 score of 42.00%. The best outcomes were not achieved by using only clean text.

Overall observations:

1. Lemmatization without stop word removal appears to improve performance for the majority of models.
2. Different models have different effects on performance when clean text (removing punctuation, HTML elements, symbols, and URLs) is applied; in some, there are no appreciable gains.
3. CleanText alone did not improve the results for any of the models.

5.1.3 Youtube_Review

In this work, Machine Learning(ML) approaches to perform sentiment analysis on the Bangla(Bengali) sentiment analysis dataset. We use four separated models Random Forest Tree, XGBOOST, NAIVE BAIYES(NB), Support Vector Machine(SVM) and results reported in this section show the effect of the three most common preprocessing techniques. In this time, the SVM model achieved the greatest accuracy with 87.38%(0.8738) and the greatest macro avg. of F1 score with 83.17% (0.8317).

In this work, we use Bangla(Bengali) sentiment analysis dataset to sentiment analysis using four ML models, Random Forest Tree, XGBOOST, NAIVE BAIYES(NB), Support Vector Machine(SVM) and find the best performance of these four models. Data preprocessing is an essential step in building a model and depending on how well the data has been preprocessed and the results are seen.These various text preprocessing steps are widely used for dimensionality reduction**.**We use three preprocessing techniques with **CleanText (Removing symbols, punctuations, html tags and urls), stop word removal(SWR) and lemmatization**. This time we do five combinations of each model and evaluate the performance of each combination.

For the Bangla(Bengali) sentiment analysis dataset, we find the best performance by SVM with cleantext as a preprocessing technique. Here, first run four models without

preprocessing.We take without preprocessing performance as a benchmark. Then find the less marginal decrease of 3%(0.03) in accuracy and 8%(0.08) in f1-score with NB from Random Forest Tree model.Also, find the most marginal increase of 1%(0.01) in accuracy and 2%(0.02) in f1-score with SVM model from Random Forest Tree model. Here, show the accuracy and f1-score of each model that is 86.61%, 85.76%, 83.06%, 87.38% and 81.94%, 81.32%, 73.99%, 83.17%.

After, run the same four models using Clean Text preprocessing. Then find the less marginal decrease of 3%(0.03) in accuracy and 9%(0.09) in f1-score with NB from Random Forest Tree model.Also, find the most marginal increase of 1%(0.01) in accuracy and f1-score with SVM model from Random Forest Tree model. Here, show the accuracy and f1-score of each model that is 86.75%, 86.07%, 83.02%, 87.38% and 82.05%, 81.74%, 73.68%, 83.17%. In this time, SVM gave the best performance over the other three models.But this time individually Random Forest Tree and XGBOOST give best performance. The whole model's best performance from the other individual model's best performance's difference is 1%(0.01). Here performance decreases in NB model, increases in Random Forest Tree and XGBOOST model and same in SVM model from the benchmark.

Afterwards, run the four models using two preprocessing techniques: Clean Text and Stop word removal. This time we got the accuracy and f1-score of each model that is 84.94%, 84.31%, 83.33%, 85.57% and 79.91%, 77.7%, 74.44%, 80.63%. Then find the less marginal decrease of 1%(0.03) in accuracy and 9%(0.09) in f1-score with NB from Random Forest Tree model. Also, find the most marginal increase of 1%(0.01) in accuracy and f1-score with the SVM model from Random Forest Tree model. This time individually NB gives the best performance. Here performance decreases in Random Forest Tree, XGBOOST, SVM model and increases in NB model from the benchmark.

Then, run the four models using two preprocessing techniques: Clean Text and Lemmatization. This time we got the accuracy and f1-score of each model that is 86.03%, 85.80%, 82.84%, 86.84% and 81.12%, 80.95%, 73.18%, 82.34%. Then find the less marginal decrease of 4%(0.04) in accuracy and 8%(0.08) in f1-score with NB from Random Forest Tree model. Also, find the most marginal increase of 0.81%(0.0081) in accuracy and 1%(0.01) in f1-score with the SVM model from Random Forest Tree model. Here performance decreases in Random Forest Tree, NB, SVM model and increases in XGBOOST model from the benchmark. Here, this model gives atrocious performance, surpassing other models.

Lastly, run the four models using three preprocessing techniques: Clean Text, Stop word removal and Lemmatization. Then find the less marginal decrease of 2%(0.02) in accuracy and 5%(0.05) in f1-score with NB from Random Forest Tree model.Also, find the most marginal increase of 1%(0.01) in accuracy and f1-score with SVM model from Random Forest Tree model. Here, show the accuracy and f1-score of each model that is 84.54%, 84.09%, 82.88%, 85.39% and 78.47%, 77.17%, 73.53%, 79.46%. This time individually Random Forest Tree and XGBOOST and SVM give the worst performance. The whole model's worst performance from the other individual model's worst performance's difference is 2%(0.0). Here all model performance decreases from the benchmark.

Since, we see three models that give the best performance using Clean text preprocessing technique. And worst performance is using Clean Text, Stop word removal and Lemmatization preprocessing technique. Here, also see most of the preprocessing technique of combination always decreases the benchmark performance.

5.2 Deep Learning

Dataset	Embedding	Models	CleanText (Removing symbols, punctuations, html tags and urls)	Stop word removal	Lemmatization	Accuracy	F1 Score
sentNob	Glove	CNN	×	×	×	0.6953	0.6540
			✓	×	×	**0.6978**	**0.6622**
			✓	✓	×	0.6716	0.6445
			✓	×	✓	0.6761	0.6438
			✓	✓	✓	0.6684	0.6382
		BiGRU	×	×	×	0.6920	0.6597
			✓	×	×	0.7106	0.6711
			✓	✓	×	0.6722	0.6524
			✓	×	✓	0.6850	0.6364
			✓	✓	✓	0.6555	0.6215
		BiLSTM	×	×	×	0.6991	0.6627
			✓	×	×	**0.7067**	**0.6777**
			✓	✓	×	0.6702	0.6415
			✓	×	✓	0.6850	0.6454
			✓	✓	✓	0.6683	0.6387
	FastText	CNN	×	×	×	0.6991	**0.6750**
			✓	×	×	**0.7049**	0.6717
			✓	✓	×	0.6690	0.6268
			✓	×	✓	0.6927	0.6614
			✓	✓	✓	0.6697	0.6236
		BiGRU	×	×	×	0.6946	0.6714
			✓	×	×	0.6959	0.6622

			✓	✓	×	0.6824	0.6424
			✓	×	✓	0.6978	0.6633
			✓	✓	✓	0.6792	0.6535
		BiLSTM	×	×	×	0.6856	0.6423
			✓	×	×	**0.7029**	**0.6668**
			✓	✓	×	0.6760	0.6473
			✓	×	✓	0.6920	0.6442
UBMEC	Glove	CNN	×	×	×	0.4880	0.4337
			✓	×	×	**0.4895**	**0.4635**
			✓	✓	×	0.4767	0.4446
			✓	×	✓	0.4854	0.4535
			✓	✓	✓	0.4707	0.4278
		BiGRU	×	×	×	0.4958	0.4709
			✓	×	×	0.5090	0.4802
			✓	✓	×	0.4816	0.4574
			✓	×	✓	**0.5109**	**0.4825**
			✓	✓	✓	0.4804	0.4589
		BiLSTM	×	×	×	0.5079	**0.4916**
			✓	×	×	0.5030	0.4853
			✓	✓	×	0.4910	0.4661
			✓	×	✓	**0.5113**	0.4832
			✓	✓	✓	0.4808	0.4576
	FastText	CNN	×	×	×	0.4673	0.4312
			✓	×	×	0.4940	**0.4626**
			✓	✓	×	0.4680	0.4328
			✓	×	✓	0.4880	0.4556
			✓	✓	✓	0.4684	0.4353
		BiGRU	×	×	×	0.5079	0.4680
			✓	×	×	**0.5270**	**0.5058**
			✓	✓	×	0.4959	0.4638
			✓	×	✓	0.5150	0.4822
			✓	✓	✓	0.4989	0.4697
		BiLSTM	×	×	×	0.5030	0.4695
			✓	×	×	0.5158	0.4940
			✓	✓	×	0.4989	0.4762
			✓	×	✓	0.5143	0.4900
	Glove	CNN	×	×	×	0.9015	0.9015
			✓	×	×	**0.9222**	**0.9019**
			✓	✓	×	0.9101	0.8861
			✓	×	✓	0.9276	0.9098
			✓	✓	✓	0.9083	0.8810
		BiGRU	×	×	×	0.9213	0.9030
			✓	×	×	0.9263	0.9089
			✓	✓	×	0.9204	0.8974
			✓	×	✓	0.9308	0.9134
			✓	✓	✓	0.9231	0.9002

YouTube_Review		BiLSTM	×	×	×	0.9312	0.9134
			✓	×	×	**0.9353**	**0.9191**
			✓	✓	×	0.9200	0.8990
			✓	×	✓	0.9339	0.9178
			✓	✓	✓	0.9231	0.9023
	FastText	CNN	×	×	×	0.9294	**0.9113**
			✓	×	×	**0.9326**	0.9171
			✓	✓	×	0.9204	0.8986
			✓	×	✓	0.9312	0.9145
			✓	✓	✓	0.9222	0.9024
		BiGRU	×	×	×	0.9285	0.9105
			✓	×	×	0.9366	0.9205
			✓	✓	×	0.9258	0.9045
			✓	×	✓	0.9389	0.9240
			✓	✓	✓	0.9213	0.9004
		BiLSTM	×	×	×	0.9330	0.9168
			✓	×	×	**0.9474**	**0.9337**
			✓	✓	×	0.9217	0.9004
			✓	×	✓	0.9402	0.9256
			✓	✓	✓	0.9258	0.9044

5.2.1 SentNoB

In this study, we explored the impact of various text preprocessing techniques on the performance of the deep learning model using sentNOB dataset. They are: BiLSTM (Bidirectional Long Short-Term Memory) with Fasttext embeddings and BiLSTM with Glove embeddings. The evaluation metrics considered include Accuracy and F1 Score.

BiLSTM and Fasttext Embedding:

The initial investigation without CleanText revealed a baseline accuracy of 68.56% and an F1 score of 64.23%. Upon the implementation of CleanText, we observed a noticeable boost, with the accuracy climbing to 70.29% and the F1 score reaching 66.68%. This underscores the significance of text cleaning in enhancing the model's ability to discern meaningful patterns.

Subsequent experiments involved the introduction of Stop Word Removal and Lemmatization. While Stop Word Removal and Clean text led to a marginal decrease in accuracy (67.60%) and F1 score (64.73%),but the combination of Stop Word Removal and Lemmatization produced the most robust performance, achieving an accuracy of 69.20% and an F1 score of 64.42%. This suggests a delicate balance between the benefits of removing common words and preserving meaningful linguistic forms.

The incorporation of all preprocessing techniques, including CleanText, Stop Word Removal, and Lemmatization, resulted in a nuanced impact, with an accuracy of 66.58% and an F1 score of 62.06%. This indicates that while comprehensive preprocessing is beneficial, there exists an interplay between techniques that requires careful consideration.

BiLSTM and Glove Embedding:

Similarly, for BiLSTM + Glove Embedding, the absence of CleanText preprocessing yielded a baseline accuracy of 69.91% and an F1 score of 66.27%. The integration of CleanText led to an improvement, showcasing an accuracy of 70.67% and an F1 score of 67.77%. This reinforces the importance of addressing textual irregularities for effective model training.

The introduction of Stop Word Removal resulted in a moderate accuracy decrease (67.02%) and F1 score reduction (64.15%). However, the combination of Stop Word Removal and Lemmatization achieved an accuracy of 68.50% and an F1 score of 64.54%, highlighting the potential trade-offs between simplicity and linguistic context preservation.

Integrating all preprocessing techniques for BiLSTM + Glove Embedding led to an accuracy of 66.83% and an F1 score of 63.87%. The findings underscore the intricate relationship between preprocessing choices and their impact on the model performance.

CNN:

In this study, we conducted sentiment analysis on the sentNob dataset using a combination of Machine Learning (ML) and Deep Learning (DL) techniques. The results revealed that DL consistently outperformed ML, with CNN achieving the highest accuracy and macro F1 of 70.49% and 67.51% respectively, compared to the 64% and 60% obtained using traditional ML techniques.

We experimented with a number of preprocessing technique combinations that were applied to the CNN model.The goal was to find the best combination that can not only be used to increase performance, but also to test whether traditional preprocessing techniques can retain the context of bengali text. Two embeddings, Fasttext and Glove, were employed, and we observed marginal differences in results, suggesting that either embedding could be used interchangeably.

We explored three preprocessing techniques: CleanText (Removing symbols, punctuations, html tags and urls), Stop word removal and Lemmatization.A total of ten combinations was tested to evaluate the impact on CNN model’s performance in Bengali Sentiment Analysis.

We first tried a run of CNN without any preprocessing to act as a benchmark. Applying only CleanText on top of either Fasttext and Glove embedding showed a marginal increase of less than 1% in accuracy and macro F1. Surprisingly, the combination of Fasttext or Glove embedding with CleanText achieved the highest performance, surpassing other preprocessing combinations. This unexpected result challenges traditional notions of preprocessing effectiveness.

Further investigations revealed that applying either lemmatization or stopword removal to the combination of CleanText and Glove/Fasttext embedding actually lowered the accuracy and F1 by around 2% each. Notably, the combination of Fasttext+CleanText+Lemmatization had the least decrease in performance from the benchmark.

However, applying all three preprocessing techniques resulted in the worst performance across both the Glove and Fasttext embedding combinations, with a decrease of 3% in accuracy and more than 3% in macro F1.

These findings suggest that traditional preprocessing techniques did not effectively capture the nuances of the Bengali language, leading to a loss of context and subsequent performance degradation across all evaluated metrics.

BIGRU:

This study investigates the impact of preprocessing techniques on the performance of Glove+BiGRU and FastText+BiGRU models using the SentNob dataset. GLOVE and FastText pre-trained word embedding models were employed, and three preprocessing techniques—CleanText, Stop Word Removal, and Lemmatization—were explored.

In the Glove+BiGRU experiments, initial results without preprocessing yielded an accuracy of 69.20% and an F1-score of 65.97%. Notably, the application of text cleaning, involving the removal of symbols, punctuations, html tags, and urls, resulted in performance enhancements, achieving an accuracy of 71.06% and an F1-score of 67.11%. However, combining text cleaning with stop word removal exhibited a decline in accuracy (67.22%) and F1 score (65.24%). CleanText coupled with lemmatization demonstrated a moderate impact, with an accuracy of 68.50% and an F1-score of 63.64%. Intriguingly, the integration of all three techniques led to a significant decrease in both accuracy and F1 score.

Turning to FastText+BiGRU, the absence of preprocessing techniques yielded an accuracy of 69.46% and an F1 score of 67.14%. Solely applying text cleaning showcased a slight improvement, with an accuracy of 69.59% and an F1-score of 66.22%. However, the combination of text cleaning and stop word removal resulted in reduced accuracy (68.24%) and F1 score (64.24%). Text cleaning and lemmatization exhibited improvement, achieving an accuracy of 69.78% and an F1-score of 66.33%. Yet, the inclusion of stop word removal and lemmatization led to a decrease in both accuracy (67.92%) and F1 score (65.35%).
In conclusion, text cleaning, specifically involving the removal of symbols, punctuations, html tags, and urls, consistently enhanced model performance. However, the incorporation of additional preprocessing techniques did not uniformly improve results and, in some instances, led to a decline in both accuracy and F1 score.

5.2.2 UMBEC

CNN:
In this work, Deep Learning (DL) approaches to perform sentiment analysis on the Unified Bangla Multi-class Emotion Corpus(UBMEC) dataset. We use two separated models CNN Glove and CNN Fasttext and results reported in this section show the effect of the three most common preprocessing techniques. In this time, CNN Fasttext model achieved the greatest accuracy with 49.4%(0.4940) and CNN Glove model achieved the greatest macro avg. of F1 score with 46.35% (0.4635).

Prior to training the CNN model, we tested with several combinations of preprocessing techniques.The objective was to determine the optimal combination that could be used to examine if conventional preprocessing approaches could preserve the context of Bengali text in addition to improving performance. We examined two embeddings, Fasttext and Glove, and found slight changes in the output, indicating that one embedding might be used in place of the other.

In this work, we use Unified Bangla Multi-class Emotion Corpus(UBMEC) dataset to sentiment analysis using two CNN models, Glove and Fasttext and find the best performance of these two models. Data preprocessing is an essential step in building a model and depending on how well the data has been preprocessed and the results are seen.These various text preprocessing steps are widely used for dimensionality reduction**.**We use three preprocessing techniques with **CleanText (Removing symbols, punctuations, html tags and urls), stop word removal(SWR) and lemmatization**. This time we do five combinations of each model and evaluate the performance of each combination.

For the UBMEC dataset, we use two CNN models Glove and Fasttext. We find the best performance by Fasttext with cleantext as a preprocessing technique. Here, first run two models without preprocessing then find the marginal decrease of 2%(0.02) in accuracy and 0.25%(0.0025) in f1-score.This time find the Glove of accuracy is 44.8%(0.4480) and f1-score is 43.37%(0.4337); Fasttext of accuracy is 46.73%(0.4673) and f1-score is 43.12%(0.4312). We take without preprocessing performance as a benchmark. After the combination we see that Fasttext models without preprocessing give atrocious performance, surpassing other preprocessing combinations.

After that, we run the Glove and Fasttext using Clean Text preprocessing and show a marginal increase of 1%(0.01) in accuracy and decrease of 0.1%(0.001) in f1-score. Here, find the accuracy of 48.95%(0.4895) in Glove and f1-score of 46.35%(0.4635); also find the accuracy of 49.40%(0.4940) and f1-score of 46.26%(0.4626) in Fasttext. Unexpectedly, outperforming other preprocessing combinations, the combination of Fasttext or Glove with CleanText produced the best results. Next, run the model using Clean Text and Stop word removal to find the decrease of 1%(0.01) in accuracy and f1-score. Here, find the accuracy of 47.67%(0.4767) in Glove and f1-score of 44.46%(0.4446); also find the accuracy of 46.80%(0.4680) and f1-score of 43.28%(0.4328) in Fasttext. Performance also decreases the benchmark in Glove model and slightly increases the Fasttext model. Next, run the model using Clean Text and Lemmatization to find the increase in accuracy and f1-score. This time, we got accuracy 48.64%(0.4864) and f1-score 45.35%(0.4535) in the Glove model ; accuracy 48.80%(0.4880) and f1-score 45.56%(0.4556) in the Fasttext model. Here, the Glove model slightly decreases and the Fasttext model slightly increases from benchmark.

Lastly, we use three preprocess techniques: Cleantext, Stop word removal and Lemmatization find a marginal decrease of 1%(0.01) in accuracy and increase of 1%(0.01) in f1-score. This time Glove gained the very worst performance that is 2%(0.02) different from the highest performance. Here, Glove model accuracy is 47.07%(0.4707) and f1-score is 42.78%(0.4278). Also, Fasttext model accuracy is 46.84% (0.4684) and f1-score is 43.53% (0.4353). This preprocessing Glove model is decreased from benchmark and Fasttext model is increased from benchmark score.

BIGRU:

In our experimentation with the UBMEC dataset, detailed in Table 1, we employed pre-trained word embedding models—specifically, GLOVE and FastText—in conjunction with a BiGRU model to assess their performance under various preprocessing techniques.

For the Glove+BiGRU model on the UBMEC dataset without text cleaning, the initial accuracy stood at 49.58%, with an F1-score of 47.09%. Upon applying the CleanText preprocessing technique, which involves removing symbols, punctuations, HTML tags, and URLs, we observed an improvement in both accuracy (50.90%) and F1 score (48.02%). Interestingly, incorporating stop word removal alongside text cleaning resulted in a decline in accuracy and F1 score (48.16% and 45.74%, respectively). The combination of CleanText and lemmatization demonstrated the most significant impact, achieving the highest accuracy of 51.09% and an F1 score of 48.25%. However, when combining all three techniques, accuracy dropped to 48.04%, and the F1 score decreased to 45.89%.

For the FastText+BiGRU model on the UBMEC dataset without preprocessing, the accuracy and F1 score were 50.79% and 46.80%, respectively. Applying CleanText led to improvements in accuracy (52.70%) and F1 score (50.58%). Combining CleanText with stop word removal resulted in a decline (49.59% accuracy and 46.38% F1 score). CleanText combined with lemmatization achieved an accuracy of 51.50% and an F1 score of 48.22%, while the combination of CleanText, stop word removal, and lemmatization yielded an accuracy of 49.89% and an F1 score of 46.97%.

These findings provide valuable insights into the impact of preprocessing techniques on the performance of embedding models. The results underscore the importance of thoughtful preprocessing in enhancing the overall performance of the models, with CleanText and lemmatization emerging as a particularly effective combination for the UBMEC dataset.

BILSTM:

In this study, we trained a BiLSTM model using two distinct word embeddings, Glove and Fasttext, while utilizing a variety of combinations of text preprocessing techniques, including CleanText (which removes symbols, punctuations, HTML tags, and URLs), lemmatization, and stop word removal. In order to create a benchmark, we also trained the model without performing any preprocessing. The objective was to ascertain the efficacy of our suggested CleanText preprocessing strategy as well as the effect of various preprocessing methods on the model's performance in terms of accuracy and F1 score (both macro and weighted average).

Our proposed CleanText preprocessing improves the model's performance by removing punctuation, HTML elements, symbols, and URLs and replacing them with specified symbols. This works particularly well when combined with Fasttext embeddings.

When compared to Glove embeddings, Fasttext embeddings often show somewhat higher accuracy and F1 scores. Interestingly, the CleanText preprocessing method improves accuracy from 50.30% to 51.58% using Fasttext embeddings, proving its efficacy. This strengthens CleanText's context-preserving feature, which is in line with Bengali language traits.

Lemmatization combined with CleanText improves the accuracy of the both Glove and Fasttext embeddings, going from 50.79% to 51.13% and 50.30% to 51.43%, respectively. When stop word removal combined with CleanText, the accuracy of the Glove and Fasttext embeddings decreases from 50.79% to 49.10% and from 50.30% to 49.89%, respectively. Lemmatization and stop word removal when combined with CleanText for Glove and Fasttext embeddings yield reductions of 2.71% and 1.54%, respectively.

Out of all the strategies studied, the combination of CleanText preprocessing with Fasttext embeddings shows the most promise, getting the top scores for accuracy, macro F1, and weighted average F1. This method, painstakingly created for Bengali text, seeks to maintain context, guaranteeing that the natural complexities and complexity of the language are preserved.

5.2.3 Youtube_Review

Therefore, we can say the optimal combination of preprocessing techniques varies between different embeddings, emphasizing the importance of tailoring preprocessing strategies to the specific requirements of the model and dataset.

5.3 Transformers

Dataset	Models	CleanText (Removing symbols, punctuations, html tags and urls)	Stop word removal	Lemmatization	Accuracy	F1 Score
SentNob	BanglaBert	×	×	×	0.7631	0.7217
		✓	×	×	0.7618	0.7280
		✓	✓	×	0.7529	0.7228

		✓	×	✓	**0.7714**	**0.7462**
		✓	✓	✓	0.7369	0.7066
	XLM-Roberta	×	×	×	0.7318	0.7026
		✓	×	×	0.7337	0.7076
		✓	✓	×	0.7266	0.6915
		✓	×	✓	**0.7394**	**0.7092**
		✓	✓	✓	0.7190	0.6843
	mBert	×	×	×	0.7157	**0.6849**
		✓	×	×	**0.7241**	0.6836
		✓	✓	×	0.7004	0.6675
		✓	×	✓	0.7061	0.6765
		✓	✓	✓	0.7042	0.6804
UBMEC	BanglaBert	×	×	×	0.6074	0.5766
		✓	×	×	0.6164	0.5943
		✓	✓	×	0.6008	0.5771
		✓	×	✓	**0.6197**	**0.5979**
		✓	✓	✓	0.5933	0.5560
	XLM-Roberta	×	×	×	0.5822	0.5591
		✓	×	×	0.5826	0.5678
		✓	✓	×	0.5500	0.5326
		✓	×	✓	0.5762	0.5464
		✓	✓	✓	0.5384	0.5109
	mBert	×	×	×	0.5586	0.5254
		✓	×	×	0.5556	0.5225
		✓	✓	×	0.5282	0.5027
		✓	×	✓	0.5544	0.5311
		✓	✓	✓	0.5282	0.5033
Youtube_Review	BanglaBert	×	×	×	0.9757	0.9695
		✓	×	×	0.9784	0.9730
		✓	✓	×	0.9667	0.9582
		✓	×	✓	0.9752	0.9692
		✓	✓	✓	0.9608	0.9510
	XLM-Roberta	×	×	×	0.9712	0.9640
		✓	×	×	0.9658	0.9570
		✓	✓	×	0.9617	0.9521
		✓	×	✓	0.9667	0.9586
		✓	✓	✓	0.9613	0.9510
	mBert	×	×	×	0.9685	0.9608
		✓	×	×	0.9672	0.9594
		✓	✓	×	0.9532	0.9412
		✓	×	✓	0.9645	0.9553
		✓	✓	✓	0.9550	0.9434

5.3.1 SentNoB

This section analyzes the performance of three Transformer models (BanglaBert, XLM-Roberta, and mBert) on the sentNob dataset for sentiment classification, considering different text preprocessing techniques.

- **BanglaBert:** As a benchmark,we obtained 76.31% accuracy and 72.17% f1-score without using any preprocessing procedures.In a rare turn of events,applying

CleanText stages (removing symbols, punctuations, HTML tags, and URLs) yielded no change, contrary to the results proved by classical ML and DL models.Further applying stop word removal did not show any improvement.

The highest results were obtained by applying lemmatization on top of cleantext, with accuracy of 77.14% and macro F1 of 74.62%. This is a rare instance of lemmatization providing the greatest results.

When cleanText, stop word removal, and lemmatization are combined, accuracy drops by 2.62% and f1-score drops by 2.45%.

- **XLM-Roberta:**As a benchmark,we obtained 73.37% accuracy and 70.26% f1-score without using any preprocessing procedures.In a rare turn of events,applying CleanText stages (removing symbols, punctuations, HTML tags, and URLs) yielded no change. Further applying stop word removal decreased the accuracy by 1%.

  The highest results were obtained by applying lemmatization on top of cleantext, with accuracy of 73.94% and macro F1 of 70.92%. This is another rare instance of lemmatization providing the greatest results.

  When cleanText, stop word removal, and lemmatization are combined, accuracy drops by 1.28% and f1-score drops by 2.33%.

- **mBert:** As a benchmark,we obtained 71.57% accuracy and 68.49% f1-score without using any preprocessing procedures.Applying CleanText (removing symbols, punctuations, HTML tags, and URLs) yielded the highest accuracy of 72.41%.

Further applying either stop word removal or lemmatization on top of cleantext decreased the accuracy and F1 by 1%.When cleanText, stop word removal, and lemmatization are combined, the results still remained lower than benchmark.

5.3.2 UMBEC

Model-wise results analysis

BanglaBert: We obtained 60.74% accuracy and 57.66% f1-score without using any preprocessing procedures, which serves as the benchmark for our comparison in this model. Lemmatization combined with the cleanText preparatory stages (removing symbols, punctuations, HTML tags, and URLs) yields the best results with an F1 score of 59.79% and an amazing accuracy of 61.97%, this particular preprocessing combination raises the

accuracy by 1.23% and the f1-score by 2.13%. Accuracy decreases when cleanText is combined with stop word removal.

When cleanText, stop word removal, and lemmatization are combined, accuracy drops by 1.41% and f1-score drops by 2.06%, whereas cleanText alone produces good results that raise both the accuracy and the f1-score.

XLM-Roberta: We obtained 58.22% accuracy and 55.91% f1-score without using any preprocessing procedures, which serves as the benchmark for our comparison in this model. For this model, cleanText preparatory stages (removing symbols, punctuations, HTML tags, and URLs) alone yields the best results with an F1 score of 56.78% and an accuracy of 58.26%. Accuracy drops by 3.22% when cleanText is combined with stop word removal.

When cleanText, stop word removal, and lemmatization are combined, accuracy drops by 4.38% and f1-score drops by 4.82%, and cleanText combined with lemmatization decreases both the accuracy and the f1-score.

mBert: We obtained 55.86% accuracy and 52.54% f1-score without using any preprocessing procedures, which serves as the benchmark for our comparison in this model. For this model, we obtained the best results with an accuracy of 55.86% without applying any preprocess techniques and with an F1 score of 53.11% by using the combination of cleanText with lemmatization.

Accuracy drops by 2.74% and f1-score drops by 2.21% when cleanText, stop word removal, and lemmatization are combined together.

General Observations

i) Among the three models (BanglaBert, XLM-Roberta, and mBert), BanglaBert consistently achieved the highest accuracy and F1 score across different preprocessing scenarios.

ii) For all three models, the inclusion of clean text (removing symbols, punctuations, HTML tags, and URLs) generally had a positive impact on performance.

iii) For BanglaBert and XLM-Roberta, either cleanText alone or combination of cleanText and lemmatization gives the highest accuracy and f1-score.

iv) The combination of clean text, stop word removal, and lemmatization for all three models consistently shows a negative impact on performance.

5.3.3 Youtube_Review

Deep Learning based Model:

Dataset	Embedding	Models	CleanText (Removing symbols, punctuations, html tags and urls)	Stop word removal	Lemmatization	Accuracy	F1 Score
YouTube_Review	Glove	CNN	✗	✗	✗	0.9015	0.9015
			✓	✗	✗	**0.9222**	**0.9019**
			✓	✓	✗	0.9101	0.8861
			✓	✗	✓	0.9276	0.9098
			✓	✓	✓	0.9083	0.8810
		BiGRU	✗	✗	✗	0.9213	0.9030
			✓	✗	✗	0.9263	0.9089
			✓	✓	✗	0.9204	0.8974
			✓	✗	✓	0.9308	0.9134
			✓	✓	✓	0.9231	0.9002
		BiLSTM	✗	✗	✗	0.9312	0.9134
			✓	✗	✗	**0.9353**	**0.9191**
			✓	✓	✗	0.9200	0.8990
			✓	✗	✓	0.9339	0.9178
			✓	✓	✓	0.9231	0.9023
	FastText	CNN	✗	✗	✗	0.9294	**0.9113**
			✓	✗	✗	**0.9326**	0.9171
			✓	✓	✗	0.9204	0.8986
			✓	✗	✓	0.9312	0.9145
			✓	✓	✓	0.9222	0.9024
		BiGRU	✗	✗	✗	0.9285	0.9105
			✓	✗	✗	0.9366	0.9205
			✓	✓	✗	0.9258	0.9045
			✓	✗	✓	0.9389	0.9240
			✓	✓	✓	0.9213	0.9004
		BiLSTM	✗	✗	✗	0.9330	0.9168
			✓	✗	✗	**0.9474**	**0.9337**
			✓	✓	✗	0.9217	0.9004
			✓	✗	✓	0.9402	0.9256
			✓	✓	✓	0.9258	0.9044

Therefore, we can say the optimal combination of preprocessing techniques varies between different embeddings, emphasizing the importance of tailoring preprocessing strategies to the specific requirements of the model and dataset.

Transformer based model:

Youtube_ Review	BanglaBert	✗	✗	✗	0.9757	0.9695
		✓	✗	✗	**0.9784**	**0.9730**
		✓	✓	✗	0.9667	0.9582
		✓	✗	✓	0.9752	0.9692
		✓	✓	✓	0.9608	0.9510
	XLM-Roberta	✗	✗	✗	**0.9712**	**0.9640**
		✓	✗	✗	0.9658	0.9570
		✓	✓	✗	0.9617	0.9521
		✓	✗	✓	0.9667	0.9586
		✓	✓	✓	0.9613	0.9510
	mBert	✗	✗	✗	**0.9685**	**0.9608**
		✓	✗	✗	0.9672	0.9594
		✓	✓	✗	0.9532	0.9412
		✓	✗	✓	0.9645	0.9553
		✓	✓	✓	0.9550	0.9434

This study delves into the nuanced effects of three fundamental preprocessing techniques CleanText, Stop Word Removal, and Lemmatization on three distinct models: BanglaBert, XLM-Roberta, and mBert. Leveraging the YouTube_Review dataset, we systematically explore the interplay between these techniques and the performance metrics of accuracy and F1 score. Through a comprehensive analysis, this study contributes valuable insights that inform the selection and customization of preprocessing strategies tailored to each model's intrinsic characteristics.

BanglaBert Model:

The BanglaBert model, renowned for its effectiveness in natural language understanding, exhibits sensitivity to preprocessing techniques. Initial experimentation without preprocessing achieved a commendable accuracy of 97.57%. The subsequent addition of CleanText alone led to marginal improvements. Intriguingly, the incorporation of Stop Word Removal alongside CleanText resulted in a notable accuracy boost to 97.84%. However, the application of Lemmatization with CleanText and Stop Word Removal led to a marginal

decline in performance, emphasizing the complex interaction between preprocessing steps for this model.

XLM-Roberta Model:

XLM-Roberta, a multilingual transformer, demonstrated resilience to unprocessed text, achieving an accuracy of 97.12%. Interestingly, the introduction of Stop Word Removal did not lead to substantial improvements. The combined application of CleanText and Stop Word Removal with Lemmatization showcased intricate dynamics, resulting in an accuracy of 96.17%. This suggests that the inherent linguistic capabilities of XLM-Roberta may mitigate the need for certain preprocessing steps.

mBert Model:

The mBert model, known for its contextualized embeddings, displayed a certain degree of resilience to uncleaned text, achieving a commendable accuracy of 96.85%. The incorporation of CleanText with Stop Word Removal further refined accuracy to 96.72%. Surprisingly, the introduction of Lemmatization, when combined with CleanText and Stop Word Removal, led to a nuanced decrease in performance, indicating that mBert may not consistently benefit from lemmatization in this context.

In summary, the results highlight the model-specific responses to preprocessing techniques, underscoring the importance of tailoring preprocessing strategies to the intrinsic characteristics of each model. These findings contribute to the ongoing discourse on effective preprocessing in natural language processing workflows.

5.4 Context-Aware-Data-Cleaning

Dataset	Model Type	Models	Features/ Embedding	Accuracy	F1 Score
UBMEC	ML	Random Forest	TF-IDF	43.13	40.57
		XGBoost		42.72	39.89
		NB		40.39	31.83
		SVM		45.16	42.24

	DL	BiLSTM	Glove	51.99	49.78
			fasttext	52.25	50.37
		BiGRU	Glove	52.25	50.41
			fasttext	53.30	51.53
		CNN	Glove	49.55	45.78
			fasttext	50.19	45.02
	Transformers	BanglaBert	Bert	63.25	61.11
		XLM-Roberta		59.16	56.99
		mBert		56.23	52.88

6. Conclusion

In conclusion, our comprehensive exploration encompassed the training of traditional machine models, diverse deep learning models employing various embeddings, and transformer-based models such as BanglaBert, mbert, and XLM-Roberta. Focusing on the Bangla language, we evaluated the impact of preprocessing techniques, including text cleaning, stop words removal, and lemmatization. Notably, our findings underscored the superior performance of Cleaned text over other preprocessing methods, surpassing the baseline model without any processing. While lemmatization contributed a marginal improvement, the removal of stop words resulted in decreased accuracy compared to the unprocessed baseline. Intriguingly, the combination of cleaned text, stop words removal, and lemmatization exhibited a further reduction in accuracy, prompting the need for continued exploration to unravel the nuanced dynamics underlying these unexpected results.

Future work for 499B

1. Eliminating manual intervention from our context-aware data-cleaning pipeline's.
2. Expanding the context-aware data-cleaning pipeline's applicability to a broader range of low-resource languages and evaluating its efficacy and performance in various linguistic contexts.

3. To assess the context-aware pipeline's applicability and effectiveness while working with ample language corpora, evaluate its scalability on bigger datasets.
4. To examine how pre-trained models might be used to enhance data cleaning and model training for low-resource languages.
5. Investigate the possibilities of transfer learning techniques within the context-aware pipeline.
6. Assess the context-aware pipeline's resilience and robustness to different levels of noise, data quality, and language diversity to make sure it will work in a variety of real-world situations.
7. Putting everything in order and finishing the project in order to submit the paper.

References

1. McRoy, S. (2021). *Principles of Natural Language Processing*. Milwaukee, WI: Susan McRoy.
2. Marlot, M., Lee, M. X., Muhammad Adib, A. I. B., Kumar Tellapaneni, P., & Lawrence, E. (2023, October). Unlocking Value from Text: Visualizing Insights with Natural Language Processing in Unstructured Oil and Gas Reports. In *SPE Asia Pacific Oil and Gas Conference and Exhibition* (p. D032S008R010). SPE.
3. Miyajiwala, A., Ladkat, A., Jagadale, S., & Joshi, R. (2022, July). On sensitivity of deep learning based text classification algorithms to practical input perturbations. In *Science and Information Conference* (pp. 613-626). Cham: Springer International Publishing.
4. Denny, M. J., & Spirling, A. (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, *26*(2), 168-189.
5. Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130-137.
6. Al-Anzi, F. S., & AbuZeina, D. (2015, December). Stemming impact on Arabic text categorization performance: A survey. In *2015 5th international conference on information & communication technology and accessibility (ICTA)* (pp. 1-7). IEEE.
7. Fortu, O., & Moldovan, D. (2005, July). Identification of textual contexts. In International and Interdisciplinary Conference on Modeling and Using Context (pp. 169-182). Berlin, Heidelberg: Springer Berlin Heidelberg.

8. Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1, 4171-4186
9. Rahman, T., & Haque, M. (2019). Bengali Language Processing: A Comprehensive Review. ACM Computing Surveys, 52(3), 1-30.
10. U.H. Hair Zaki, R. Ibrahim, S. Abd Halim, I.I. Kamsani, Text detergent: The systematic combination of text preprocessing techniques for social media sentiment analysis, in: International Conference of Reliable Information and Communication Technology, Springer, 2022, pp. 50–61.TY - CONF
11. Singh, T., & Kumari, M. (2016). Role of text preprocessing in twitter sentiment analysis. *Procedia Computer Science*, *89*, 549-554.
12. Srivastava, A., Makhija, P., & Gupta, A. (2020, November). Noisy text data: Achilles' heel of BERT. In Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020) (pp. 16-21).
13. Sun, Y., & Jiang, H. (2019). Contextual text denoising with masked language models. arXiv preprint arXiv:1910.14080.
14. Khan, J., & Lee, S. (2021). Enhancement of Text Analysis Using Context-Aware Normalization of Social Media Informal Text. Applied Sciences, 11(17), 8172
15. Siino, M., Tinnirello, I., & La Cascia, M. (2023). Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers. *Information Systems*, 102342.
16. 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.
17. Islam, K. I., Kar, S., Islam, M. S., & Amin, M. R. (2021, November). SentNoB: A dataset for analysing sentiment on noisy Bangla texts. In Findings of the Association for Computational Linguistics: EMNLP 2021 (pp. 3265-3271).
18. Sakib Ullah Sourav, M., Wang, H., Sultan Mahmud, M., & Zheng, H. (2022). Transformer-based Text Classification on Unified Bangla Multi-class Emotion Corpus. arXiv e-prints, arXiv-2210.
19. Islam, K. I., Yuvraz, T., Islam, M. S., & Hassan, E. (2022, November). EmoNoBa: A Dataset for Analyzing Fine-Grained Emotions on Noisy Bangla Texts. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (pp. 128-134).
20. Md Zobaer Hossain, Md Ashraful Rahman, Md Saiful Islam, and Sudipta Kar. 2020. BanFakeNews: A Dataset for Detecting Fake News in Bangla. In Proceedings of the

Twelfth Language Resources and Evaluation Conference, pages 2862–2871, Marseille, France. European Language Resources Association

21. (2011). TF–IDF. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_832
22. Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).
23. Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, 135-146.
24. Kenton, J. D. M. W. C., & Toutanova, L. K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of naacL-HLT (Vol. 1, p. 2).
25. Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. IEEE Intelligent Systems and their applications, 13(4), 18-28.
26. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
27. Webb, Geoffrey I., Eamonn Keogh, and Risto Miikkulainen. "Naïve Bayes." Encyclopedia of machine learning 15, no. 1 (2010): 713-714.
28. Webb, Geoffrey I., Eamonn Keogh, and Risto Miikkulainen. "Naïve Bayes." Encyclopedia of machine learning 15, no. 1 (2010): 713-714.
29. Fukushima, K. (2007). Neocognitron. Scholarpedia, 2(1), 1717.
30. Memory, L. S. T. (2010). Long short-term memory. Neural computation, 9(8), 1735-1780.
31. Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673-2681.
32. Bhattacharjee, A., Hasan, T., Ahmad, W. U., Samin, K., Islam, M. S., Iqbal, A., ... & Shahriyar, R. (2021). BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla. arXiv preprint arXiv:2101.00204.
33. Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555.
34. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

35. Doddapaneni, S., Aralikatte, R., Ramesh, G., Goyal, S., Khapra, M. M., Kunchukuttan, A., & Kumar, P. (2023, July). Towards leaving no Indic language behind: building monolingual corpora, benchmark and models for Indic languages. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 12402-12426).
36. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116.
37. Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.
38. Prechelt, L. (2002). Early stopping-but when?. In Neural Networks: Tricks of the trade (pp. 55-69). Berlin, Heidelberg: Springer Berlin Heidelberg.
39. Smith, L. N. (2017, March). Cyclical learning rates for training neural networks. In 2017 IEEE winter conference on applications of computer vision (WACV) (pp. 464-472). IEEE.