**A Comprehensive Rule-Based Approach for Bengali Text Normalization Addressing Language-Specific Challenges**

**Abstract**: Text normalization, the conversion of non-standard textual forms into a standardized representation, is indispensable for robust Natural Language Processing (NLP) and Speech Technology. While methodologies exist for resource-rich languages, applying them to Bengali (Bangla) necessitates addressing unique linguistic and orthographic complexities. These include the concurrent use of Bengali and Western digits, a distinct numerical scaling system (crore, lakh), multifaceted ordinal expressions, considerable variation in date, time, and currency formats, and the pervasive phenomenon of code-mixing with English. This paper details a comprehensive rule-based system meticulously designed to tackle these Bengali-specific normalization challenges. Our system employs a modular pipeline architecture integrating sophisticated pattern identification using regular expressions with curated linguistic knowledge bases and procedural transformation logic. It systematically verbalizes cardinals, ordinals, temporal expressions, monetary values (Taka), percentages, temperatures, ratios, and phone numbers, standardizing them into spoken Bengali forms. Furthermore, it incorporates a strategy for handling English loanwords via phonetic transliteration and includes a module for generating International Phonetic Alphabet (IPA) representations, vital for speech synthesis applications. We elaborate on the system’s architecture, the linguistic rationale behind its rules, and its approach to handling Bengali’s specific characteristics. Through illustrative case studies, we demonstrate its capability. Crucially, we engage in a comparative discussion regarding rule-based versus machine learning (ML) approaches, arguing that for Bengali, given the current resource landscape and the need for linguistic precision and interpretability, a well-engineered rule-based system presents a highly effective, pragmatic, and resource-efficient solution, challenging the notion that complex ML models are invariably superior for all normalization tasks.

**Keywords:** Bengali Text Normalization, Bangla Text Processing, Rule-Based Systems, Text Standardization, Natural Language Processing, Speech Synthesis Preprocessing, IPA Conversion, Language-Specific NLP, Code-Mixing Handling.

**1. Introduction**

The digital revolution has led to an explosion of textual data across languages, but much of this text exists in forms unsuitable for direct computational processing. Raw text often contains non-standard words (NSWs) – numerals, abbreviations, symbols, dates, times – that introduce ambiguity and inconsistency. Text normalization addresses this by converting these NSWs into a uniform, typically written-out or spoken-word, format [1, Placeholder Reference]. This process is not merely cosmetic; it is a critical preprocessing step that significantly enhances the performance and reliability of downstream applications like Text-to-Speech (TTS) synthesis [2, Placeholder Reference], Automatic Speech Recognition (ASR) [Placeholder Reference], Machine Translation (MT) [Placeholder Reference], Information Extraction, and Sentiment Analysis. Consistent textual representation reduces vocabulary sparsity, simplifies language modeling, and ensures accurate interpretation or rendering of the text’s meaning.

While the principles of text normalization are universal, their practical implementation is highly language-dependent. Systems designed for English often fail when applied directly to languages with different scripts, morphology, or conventions [Placeholder Reference]. Bengali, a major world language with a rich literary tradition and burgeoning digital presence, exemplifies this need for language-specific solutions. Normalizing Bengali text involves navigating a unique confluence of challenges:

* **Orthographic Diversity:** Bengali script coexists with the Latin script in digital communication. Notably, both native Bengali digits (০, ১, ২, …, ৯) and Western digits (0, 1, 2, …, 9) are widely used, sometimes interchangeably, demanding systems capable of recognizing and processing both [Placeholder Reference].
* **Unique Numerical System:** The Bengali number system incorporates traditional South Asian scaling units – কোটি (crore: 107), লক্ষ (lakh: 105), হাজার (hazar: 103) – alongside শত (sho: 102). Normalization must correctly segment numbers and verbalize them according to this system [Placeholder Reference].
* **Complex Ordinalization:** Expressing order involves not just simple suffixes but a variety of morphological endings (e.g., -ম, -য়, -লা, -রা, -শে, -ই, -তম) and specific lexical items (e.g., প্রথম ‘first’, দ্বিতীয় ‘second’), requiring extensive linguistic knowledge for correct verbalization [Placeholder Reference].
* **Formatting Conventions:** Dates, times, currency (the Bangladeshi Taka, ৳), percentages, and other units are written in numerous formats using varying delimiters (/, -, .), Bengali or English month names, and language-specific keywords (e.g., টাকা, শতাংশ, ডিগ্রি, টায়, সাল).
* **Pervasive Code-Mixing:** Integrating English words – technical terms, named entities, social media jargon – into Bengali sentences is common, especially online. A normalization system must decide how to handle these foreign lexical items, often opting for phonetic transliteration rather than translation [Placeholder Reference].

Addressing these challenges comprehensively is essential for advancing Bengali NLP. The dominant trend in modern NLP favors Machine Learning (ML) approaches, particularly deep learning models [8, 9, Placeholder Reference]. However, these models typically demand vast quantities of labeled training data (parallel corpora of raw and normalized text), which remain scarce for Bengali text normalization. Furthermore, ensuring linguistic accuracy and consistency for specific rule-governed phenomena (like number scaling or date formats) can be difficult with purely data-driven methods, and their lack of inherent interpretability can hinder debugging and trust [10, Placeholder Reference].

This paper presents an alternative yet powerful approach: a comprehensive, rule-based system meticulously crafted for Bengali text normalization. We posit that a well-designed system grounded in linguistic knowledge and explicit rules offers a compelling solution, particularly considering the specific challenges of Bengali and the current data landscape. Our system utilizes a modular pipeline, combining robust pattern identification via regular expressions with extensive linguistic mapping dictionaries and procedural conversion algorithms.

The primary contributions of this research are:

1. A detailed account of a comprehensive rule-based system addressing a wide spectrum of NSW categories specific to the Bengali language.
2. In-depth exploration of rule-based strategies tailored for Bengali linguistic features, including its unique numbering system, ordinal variations, date/time conventions, and orthographic diversity.
3. An implemented solution for code-mixing using phonetic transliteration of common English loanwords within a Bengali context.
4. The integration of a Bengali-to-International Phonetic Alphabet (IPA) conversion component, facilitating direct use in speech synthesis pipelines.
5. A reasoned argument, supported by the system's design and capabilities, for the effectiveness, interpretability, and pragmatic value of the rule-based paradigm for tackling Bengali text normalization challenges, presenting it as a robust alternative or complement to purely ML-driven methods.

The structure of this paper is as follows: Section 2 reviews related work in text normalization, covering different methodologies and efforts for Indic languages. Section 3 provides a detailed description of our system’s architecture and the underlying methodologies for pattern identification, verbalization, and handling specific linguistic phenomena. Section 4 presents illustrative case studies demonstrating the system’s functionality. Section 5 engages in a critical discussion comparing the rule-based approach with potential ML solutions, analyzing the trade-offs in the context of Bengali. Finally, Section 6 concludes the paper and proposes directions for future research. The system’s codebase is made available for research purposes at: [[**HERE**](https://github.com/shohanur-shoron/Bangla-Cleaner.git)].

**2. Related Work**

Text normalization has been a subject of research and development for decades, driven primarily by the needs of TTS and ASR systems. Methodologies have evolved, but core challenges related to ambiguity and coverage persist across languages.

**Rule-Based and Finite-State Approaches:** Historically, rule-based systems have been the mainstay of text normalization [1, 3]. These systems often employ cascades of hand-crafted rules, implemented using regular expressions or, more formally, Finite-State Transducers (FSTs) [4, 5]. FSTs provide a powerful and efficient mechanism for defining complex rewrite rules and have been successfully applied to morphological analysis and normalization tasks in various languages [Placeholder Reference]. Companies like Google and AT&T developed sophisticated rule-based normalization systems based on FSTs, demonstrating their scalability and effectiveness for commercial TTS systems [10, Placeholder Reference]. Their strength lies in precision and the ability to encode explicit linguistic knowledge. However, developing and maintaining large rule sets requires significant expertise and can be brittle when faced with unexpected input variations.

**Normalization for Indic Languages:** Research specific to Indic languages, including Bengali, often highlights the necessity of adapting normalization techniques to handle unique script properties, complex morphology, and distinct numerical systems [6, 7, Placeholder Reference]. Work on Hindi, Malayalam, Tamil, and other languages has explored rule-based and hybrid methods to address challenges like consonant conjuncts, agglutination, variations in date/time formats, and the lakh/crore numbering convention [Placeholder Reference]. However, comprehensive, publicly detailed systems covering the full spectrum of normalization categories specifically for Bengali remain relatively uncommon.

**Machine Learning Approaches:** The rise of deep learning has introduced sequence-to-sequence (Seq2Seq) models, including those based on Recurrent Neural Networks (RNNs) and Transformers, to the text normalization task [8, 9, Placeholder Reference]. These models learn the mapping from input NSW sequences to their normalized output sequences directly from large parallel corpora. They have shown impressive results for English, demonstrating an ability to handle diverse contexts and potentially generalize better to unseen patterns than purely rule-based systems [Placeholder Reference]. However, their performance is heavily contingent on the availability of substantial, high-quality training data. Creating such datasets for languages like Bengali is a significant bottleneck. Moreover, these models can sometimes produce linguistically implausible outputs or struggle with maintaining consistency for specific numerical or grammatical rules without sophisticated training strategies or architectural modifications [Placeholder Reference]. Hybrid approaches, combining ML with rule-based components or FSTs, have also been explored to leverage the strengths of both paradigms [Placeholder Reference].

**Positioning the Current Work:** Our work contributes a detailed description and implementation of a comprehensive *rule-based* system specifically for Bengali. While acknowledging the advancements in ML, we focus on demonstrating the power and practicality of a linguistically informed, rule-driven approach, particularly relevant given Bengali’s specific challenges and the current data availability landscape. We emphasize modularity, interpretability, and coverage of Bengali-specific phenomena, including phonetic conversion, positioning the system as a valuable tool and a strong baseline for future comparative studies involving ML techniques for Bengali text normalization.

**3. System Architecture and Methodology**

Our Bengali text normalizer is built upon a modular pipeline architecture designed for clarity, maintainability, and controlled execution of normalization rules. The system processes input text sequentially through distinct functional components, ensuring that specific patterns are handled in a logical order to resolve potential ambiguities and maximize accuracy.

**3.1. Overall Design: The Modular Pipeline Philosophy**

The core principle is to break down the complex task of normalization into manageable sub-tasks, each handled by a dedicated component. The main normalization function orchestrates the flow, invoking these components in a predefined sequence. This pipeline approach is critical because the correct interpretation of a token often depends on context and the potential for it to belong to multiple categories. For instance:

* **Specificity Order:** Highly specific patterns (like dates “DD/MM/YYYY” or currency “৳XXX”) must be identified and processed before more general patterns (like standalone numbers “DD” or “XXX”) to avoid partial or incorrect normalization.
* **Context Preservation:** The system aims to normalize NSWs in place, preserving the surrounding textual context.
* **Robustness:** For longer texts, processing sentence-by-sentence within error-handling blocks prevents the entire normalization process from failing due to an issue in a single problematic segment.

The main functional stages within the pipeline are: 1. Identification of potential NSWs based on predefined patterns. 2. Verbalization or transformation of identified NSWs into standard Bengali word forms. 3. Handling of specific linguistic categories (numerals, ordinals, temporals, etc.). 4. Processing of code-mixed elements (English loanwords). 5. Optional phonetic conversion to IPA. 6. Final cleanup of formatting artifacts (e.g., extra spaces).

**3.2. Pattern Identification Engine**

This component is responsible for detecting potential NSW candidates within the input text. Instead of relying on simple keyword spotting, it employs sophisticated pattern matching using regular expressions. Key aspects of this engine include:

* **Regular Expression Power:** Carefully crafted regex patterns are defined for each NSW category (dates, times, numbers, currency, percentages, temperatures, ratios, ordinals, phone numbers, years). These patterns are designed to be specific enough to avoid false positives while being flexible enough to capture common variations in formatting and script usage ([০-৯0-9] for digits, handling various separators like -, /, :, etc.).
* **Contextual Boundary Enforcement:** Crucial use of regex lookarounds ((?<=...), (?=...), (?!...)) ensures that patterns are matched only when they represent a distinct NSW unit. For example, a sequence of digits is identified as a standalone number only if not immediately preceded or followed by characters that would make it part of another word, code, or a different type of NSW already handled (like a date).
* **Capturing Variations:** Patterns account for optional elements (e.g., seconds in time, currency symbols, percentage signs vs. words like “শতাংশ”) and alternative representations (e.g., different month formats, AM/PM variations).

This pattern identification engine provides the input for the subsequent verbalization stage.

**3.3. Verbalization and Transformation Core**

Once an NSW pattern is identified, this core component handles its conversion into the standard Bengali word form. This involves more than simple lookups; it combines algorithmic logic with linguistic knowledge.

* **Algorithmic Conversion:** Procedural logic is implemented for tasks requiring computation or structural manipulation:
  + *Numerical Scaling:* A dedicated algorithm converts numeric values into words according to the Bengali crore/lakh/hazar/sho system, recursively handling large numbers by segmenting them appropriately.
  + *Decimal Verbalization:* Numbers with decimal points are split; the integer part is verbalized using the scaling algorithm, and the fractional part is typically read out digit by digit following the word “দশমিক” (doshomik - decimal).
  + *Date/Time Assembly:* Components extracted from date/time patterns (day, month, year, hour, minute, second) are individually converted to words and then assembled grammatically, including necessary particles or suffixes (e.g., “টা” for hour, “মিনিটে” for minute) and determining the appropriate time period (e.g., “সকাল”, “দুপুর”, “রাত”). Date parsing leverages robust external libraries where appropriate, after initial preprocessing to handle Bengali specifics.
* **Linguistic Knowledge Base:** The conversion heavily relies on curated dictionaries storing mappings:
  + Digit-to-word maps for both Bengali and English digits (0-99).
  + Month name correspondences (Bengali to standard English for parsing, and potentially Bengali month names directly for verbalization).
  + Ordinal mapping tables covering common suffixes and lexical ordinals.
  + Phonetic mappings for English words and Bengali characters/conjuncts to IPA.
  + Standard suffixes for units (“শতাংশ”, “ডিগ্রি সেলসিয়াস”, “টাকা”).
* **Utility Support:** Helper functions provide essential services like inter-script digit translation, string cleaning, and parsing support.

**3.4. Handling Specific Linguistic Categories**

The system incorporates specialized modules or logic streams within the pipeline to handle the nuances of different NSW types in Bengali:

* **Cardinal Numbers:** Verbalizes integers and floats, correctly applying the crore/lakh/hazar/sho scaling, handling decimal points, and recognizing negative signs. Accommodates both Bengali and Western input digits.
* **Temporal Expressions (Dates & Times):** Parses a wide variety of formats. Converts day numbers using appropriate Bengali ordinal forms for dates (e.g., “পহেলা”, “দোসরা”, “একুশে”). Handles Bengali and English month names. Constructs time expressions including the period of the day (সকাল, দুপুর, etc.) and relevant suffixes (“টা”, “মিনিট”, “সেকেন্ড”).
* **Monetary Values (Taka):** Specifically identifies patterns involving the Taka symbol (৳) or the word টাকা/টাকার. It isolates the numerical value, including potential units like লক্ষ or কোটি appearing after the number, verbalizes the number using the standard algorithm, and reconstructs the phrase correctly (e.g., “৳১০ লক্ষ” -> “দশ লক্ষ টাকা”).
* **Percentages:** Recognizes numerical values followed by % or শতাংশ. Verbalizes the number and appends the standard word “শতাংশ”.
* **Ordinals:** Addresses the complexity of Bengali ordinals by using an extensive mapping for common forms (১ম, ২য়, ৩রা, ৪ঠা, ৫ই, ২১শে, ২২তম, 1st, 2nd etc.) to their full word equivalents (প্রথম, দ্বিতীয়, তেসরা, চৌঠা, পাঁচই, একুশে, বাইশতম, প্রথম, দ্বিতীয়). For numbers followed by generic suffixes like -তম that are not in the map, it verbalizes the cardinal number and appends the appropriate ordinal term.
* **Ratios:** Identifies formats like X:Y, XঃY, X-Y, X থেকে Y. It typically replaces symbolic separators with appropriate Bengali wording (e.g., : becomes এ or similar phrasing indicating ratio) and verbalizes the constituent numbers.
* **Specialized Numbers:** Includes dedicated logic for phone numbers (read digit-by-digit, handling ‘+’), temperatures (parsing value and units like °C, °F, ডিগ্রি, converting number, appending full unit name like “ডিগ্রি সেলসিয়াস”), and context-dependent year expressions (verbalizing four-digit years found near “সাল” or “সন”).

**3.5. Addressing Code-Mixing: Phonetic Transliteration Strategy**

Recognizing the prevalence of English words in modern Bengali text, the system incorporates a module for handling code-mixing. Instead of attempting direct translation (which is often inappropriate for proper nouns, technical terms, or widely adopted loanwords), it employs phonetic transliteration.

* **Rationale:** For words like “Google,” “Facebook,” “computer,” or “internet,” a Bengali speaker would typically pronounce them using Bengali phonetics rather than translating them to a Bengali equivalent (which might not even exist or sound natural).
* **Mechanism:** A curated dictionary maps common English words (case-insensitive) to their widely accepted Bengali phonetic spellings (e.g., computer -> কম্পিউটার). The system iterates through the text, identifies words present in this map, and replaces them with their Bengali phonetic counterpart.
* **Scope and Limitations:** This approach focuses on high-frequency loanwords. It does not attempt to transliterate arbitrary English words but rather handles a predefined, common vocabulary to improve the naturalness of normalized text intended for reading or speech synthesis.

**3.6. Bengali Phonetic Conversion to IPA**

To directly support speech synthesis front-ends or phonetic analysis, the system includes a component to convert the (already normalized) Bengali text into the International Phonetic Alphabet (IPA).

* **Importance for Speech:** IPA provides an unambiguous representation of pronunciation, crucial for TTS systems to generate accurate sounds.
* **Mapping Strategy:** It uses two levels of mapping:
  + *Individual Characters:* Maps basic Bengali vowels, consonants, and diacritics to their corresponding IPA symbols.
  + *Consonant Conjuncts (যুক্তাক্ষর):* Utilizes an extensive dictionary mapping Bengali conjuncts (like ক্ক, ক্ষ, জ্ঞ, ন্ত, ম্প্র, শ্চ etc.) to their specific IPA pronunciations. This is critical because conjunct pronunciation often deviates significantly from simply concatenating the IPA of constituent consonants.
* **Longest Match Principle:** The conversion algorithm prioritizes matching the longest possible conjunct sequence at any point in the text before falling back to individual character mapping, ensuring accurate representation of complex clusters.

**3.7. The Role of Linguistic Data Resources**

The foundation of this rule-based system lies in the curated linguistic knowledge encoded in its data files. This includes:

* Comprehensive number-to-word mappings.
* Mappings for ordinals, months, and days.
* The English-to-Bengali phonetic transliteration dictionary.
* Detailed IPA mappings for individual characters and, crucially, for a wide range of consonant conjuncts.

The accuracy and coverage of the normalizer are directly tied to the quality and extensiveness of these resources. Their creation represents a significant knowledge engineering effort, capturing specific linguistic rules and conventions of Bengali.

**3.8 Computational Complexity and Resource Considerations**

An important aspect of evaluating any text processing system is understanding its computational demands and how its performance scales with input size.

**Complexity of the Rule-Based System:** The normalization process implemented in our system operates as a pipeline with a fixed number (*P*) of distinct stages (pattern extraction, conversion, replacement, etc.). Within each stage, the primary operations involve regular expression matching (re.findall) over the text chunk (length *L*) and subsequent string manipulations or replacements based on the matches found (*M*). While regex matching can have theoretical worst cases, for the patterns employed here, practical performance is typically linear, approximately **O(L)**. Sorting matches takes O(M log M), usually negligible as M << L. The conversion and replacement steps also largely depend linearly on *L* or the length of the matches. Therefore, a single stage operates in roughly O(L) time.

Since the pipeline has a constant number of stages *P*, the overall time complexity for processing an input text of length *N* (whether as one chunk or *S* chunks of average length *N/S*) is approximately *P* \* O(N/S) \* *S* = **O(N)**. The optional IPA conversion adds another linear pass, maintaining the overall **linear time complexity** relative to the input text length. This linear scaling means that processing time grows predictably and proportionally with the size of the text.

**Comparison with Machine Learning Inference Complexity:** This linear complexity contrasts favorably with the theoretical inference time complexities of common ML models used for sequence-to-sequence tasks:

* **RNNs (LSTMs/GRUs):** These often exhibit complexities involving terms like O(N \* M) or higher, influenced by hidden state sizes (*d*) and attention mechanisms, though still broadly linear in sequence lengths.
* **Standard Transformers:** The dominant architecture currently features a self-attention mechanism leading to a theoretical time complexity of **O(N² \* d)** for the encoder, primarily due to the N² term related to input length *N*. While optimized variants aim to reduce this, the standard Transformer's quadratic scaling can become a bottleneck for very long sequences.

**Resource Considerations and Practical Implications:** Beyond theoretical complexity, practical performance and resource requirements differ significantly:

* **Computational Intensity:** Our rule-based system relies on CPU-bound operations like string matching, dictionary lookups, and basic procedural logic. These operations are generally computationally lightweight compared to the massive matrix multiplications involved in deep learning models.
* **Hardware Dependencies:** ML models, especially large Transformers, achieve practical speed through massive parallelism on specialized hardware like GPUs or TPUs. Their performance is often orders of magnitude slower on standard CPUs. Our rule-based system, being primarily CPU-bound, runs efficiently on standard, widely available hardware, including systems with limited computational resources. **It does not require specialized accelerators**, making it highly accessible and deployable even on very low-resource or embedded systems where running large ML models would be infeasible due to power, memory, or computational constraints.
* **Predictability vs. Parallelism:** While the rule-based system offers predictable O(N) scaling on CPUs, ML models on GPUs can achieve higher throughput for moderate sequence lengths due to parallelism, despite potentially worse asymptotic complexity. However, the rule-based system's lower overhead and lack of dependence on specialized hardware make it a more resource-efficient choice in many deployment scenarios.

The proposed rule-based system offers efficient **O(N) time complexity** and importantly, **low resource requirements**, allowing it to run effectively on standard CPUs and resource-constrained environments. This contrasts with typical ML models which, while potentially faster on specialized hardware for certain workloads, have higher theoretical complexity (often O(N²)) and significantly greater computational and hardware demands.

**4. Experiments and Case Studies**

Evaluating the performance of a text normalization system ideally requires standardized benchmark datasets and metrics. However, such resources are not readily available for Bengali normalization. In their absence, we demonstrate the system’s functionality and effectiveness through qualitative analysis based on a diverse set of crafted test sentences designed to exercise different aspects of the normalization pipeline.

**4.1. Implementation Details**

The system is implemented using Python 3.x. It leverages the standard re library for regular expressions and dateutil.parser for flexible date string interpretation (after preprocessing Bengali elements). The code is structured into modular components reflecting the architecture described in Section 3.

**4.2. Test Data Description**

The test suite comprises sentences containing various combinations of NSWs, including:

* Dates and times in multiple formats (DD/MM/YYYY, DD-Month-YYYY, YYYY-MM-DD, HH:MM, HH:MM:SS, AM/PM).
* Numbers of varying magnitudes, including decimals and the crore/lakh system.
* Currency amounts with the ৳ symbol, টাকা/টাকার keywords, and scaling units.
* Percentages using % and শতাংশ.
* Ordinals using Bengali and English suffixes.
* Temperatures with different units (°C, °F, ডিগ্রি).
* Ratios with different separators (:, ঃ, -).
* Phone numbers with national prefixes.
* Code-mixed English words.
* Contextual year expressions.

**4.3. Qualitative Analysis: Illustrative Examples**

We present examples showcasing the transformation from raw input to normalized Bengali text, followed by the generated IPA string.

* **Example 1: Basic Numerals, Date, Time**
  + *Input:* গতকাল ১৯শে মে, ২০২৩ এ দুপুর ২:৩০ মিনিটে ৩টি বই কিনেছি।
  + *Normalized Output:* গতকাল উনিশে মে দুই হাজার তেইশ এ দুপুর দুই টা ত্রিশ মিনিটে তিনটা বই কিনেছি
  + *IPA Output:* ɡɔt̪okaːl uniʃe me d̪ui haːdʒaːr t̪eiʃ ɛ d̪upur d̪ui ʈaː t̪riʃ miniʈe t̪inʈaː boi kinetʃʰi
  + *Analysis:* Correctly handles the ordinal date (“উনিশে মে”), year (“দুই হাজার তেইশ”), time with period (“দুপুর দুই টা ত্রিশ মিনিটে”), and simple cardinal (“তিনটা”).
* **Example 2: Currency with Scaling, Percentage, Temperature**
  + *Input:* দাম ৳১.৫ লক্ষ, ছাড় ৫০%, তাপমাত্রা ছিল -৫°C।
  + *Normalized Output:* দাম এক দশমিক পাঁচ লক্ষ টাকা ছাড় পঞ্চাশ শতাংশ তাপমাত্রা ছিল মাইনাস পাঁচ ডিগ্রি সেলসিয়াস
  + *IPA Output:* d̪aːm ɛk d̪ɔʃomik paːtʃ lokkʰo ʈaːkaː tʃʰaːɽ pɔntʃaːʃ ʃɔt̪aːŋʃo t̪aːpmːraː tʃʰilo mainaːs paːtʃ ɖigri selsius
  + *Analysis:* Successfully processes currency with “লক্ষ”, includes decimal verbalization, handles percentage, and correctly verbalizes negative temperature with the appropriate unit.
* **Example 3: Ordinal (English Suffix), Ratio, Phone Number, Code-Mixing**
  + *Input:* আমার 1st Rank ছিল, অনুপাত 3:1, ফোন +8801710000000, Google সেরা।
  + *Normalized Output:* আমার প্রথম Rank ছিল অনুপাত তিন এ এক ফোন প্লাস আট আট শূন্য এক সাত এক শূন্য শূন্য শূন্য শূন্য শূন্য শূন্য গুগল সেরা
  + *IPA Output:* aːmaːr prot̪ʰom Rank tʃʰilo onupː t̪in ɛ ɛk pʰon plaːs aːʈ aːʈ ʃunno ɛk ʃaːt̪ ɛk ʃunno ʃunno ʃunno ʃunno ʃunno ʃunno ɡuɡɔl ʃeraː
  + *Analysis:* Maps the English ordinal suffix “1st” to “প্রথম”, handles the ratio, reads the phone number digit-by-digit including “প্লাস”, and transliterates “Google” to “গুগল”. (Note: “Rank” is left as is, assuming it’s not in the transliteration map).
* **Example 4: Complex Date Format, Conjuncts in IPA**
  + *Input:* ঘটনাটি ঘটেছিল ২৫-ডিসেম্বর-২০২৪ তারিখে, যা অত্যন্ত গুরুত্বপূর্ণ।
  + *Normalized Output:* ঘটনাটি ঘটেছিল পঁচিশে ডিসেম্বর দুই হাজার চব্বিশ তারিখে যা অত্যন্ত গুরুত্বপূর্ণ
  + *IPA Output:* gʰɔʈonaːʈi gʰoʈetʃʰilo pɔ̃tʃiʃe ɖisembor d̪ui haːdʒaːr tʃobbiʃ t̪aːrikʰe dʒaː ɔt̪t̪ɔnt̪o ɡurut̪t̪opurno
  + *Analysis:* Parses the complex date format. The IPA output correctly reflects conjuncts like “ন্ত” (nt̪) in “অত্যন্ত” and “র্ত” (rt̪) in “গুরুত্বপূর্ণ”.

These examples illustrate the system’s capacity to handle diverse inputs according to predefined linguistic rules, producing standardized and phonetically plausible outputs.

**4.4. Discussion on Evaluation Metrics and Challenges for Bengali**

While qualitative examples demonstrate functionality, rigorous evaluation is paramount. Standard metrics include accuracy (requiring annotated data), WER/CER reduction (for ASR integration), and MOS scores (for TTS integration) [Placeholder Reference]. The primary obstacle for Bengali text normalization is the **lack of standardized, publicly available benchmark datasets**. Creating such a resource is non-trivial, requiring significant annotation effort to cover the vast range of NSWs, contexts, and potential ambiguities present in real-world Bengali text (from formal documents to informal social media). Establishing community agreement on the “correct” normalized form for all cases can also be challenging. Without such benchmarks, comparing different normalization systems quantitatively remains difficult. Our current evaluation focuses on demonstrating comprehensive coverage and logical correctness through carefully chosen examples.

**5. Discussion: The Case for Rule-Based Normalization in Bengali**

The development and performance of our system prompt a broader discussion about the most suitable approach for Bengali text normalization in the current technological and resource context. While ML models dominate many NLP tasks, we argue that a well-crafted rule-based system offers compelling advantages for this specific challenge.

**5.1. Advantages of the Rule-Based Paradigm Revisited**

* **Interpretability and Trust:** Every output of our system can be explained by tracing the execution path through specific rules, regex matches, and dictionary lookups. This transparency is crucial for understanding failures, debugging edge cases, and building user trust, especially in applications like TTS where errors can be jarring. This contrasts significantly with the “black-box” nature of many complex neural models [10].
* **Linguistic Precision and Control:** Bengali normalization involves adhering to specific grammatical and numerical conventions (e.g., crore/lakh structure, exact ordinal forms, date component ordering). Rules allow these conventions to be encoded explicitly and enforced reliably. Achieving this level of granular control with purely data-driven models can require immense amounts of very specific training data or complex model architectures with built-in constraints [Placeholder Reference].
* **Data Efficiency:** The development relied on linguistic expertise and curated knowledge bases, not massive parallel corpora. This makes the rule-based approach significantly more feasible for languages like Bengali where large-scale, high-quality labeled datasets for normalization are not readily available. It lowers the barrier to entry for creating effective normalization tools.
* **Handling Well-Defined Phenomena:** Rule-based systems excel at handling phenomena that follow predictable patterns, even if complex. Dates, times, standardized currency formats, and the core number system fall into this category. The system provides consistent and accurate output for these crucial NSW types.

**5.2. Acknowledging the Limitations**

It is equally important to acknowledge the inherent limitations of the rule-based approach:

* **Brittleness to Variation:** The system’s performance degrades when faced with patterns significantly different from those encoded in the rules. Highly informal language, creative misspellings, novel abbreviations, or completely unforeseen formatting will likely lead to errors or missed normalizations. ML models, potentially, offer better robustness to such noise and variation *if* trained on sufficiently diverse data.
* **Scalability and Maintenance Effort:** As the desire for broader coverage increases, the number and complexity of rules grow. Maintaining this rule set, ensuring consistency, and avoiding conflicts requires ongoing expert effort and rigorous testing. Adding new rules can have unintended consequences elsewhere in the pipeline.
* **Manual Knowledge Engineering:** The initial development and subsequent refinement require significant manual effort in linguistic analysis, rule crafting, and dictionary curation. This contrasts with the ML paradigm where effort shifts towards data collection, annotation, and model training/tuning.

**5.3. Rule-Based vs. Machine Learning: A Pragmatic Perspective for Bengali**

The choice between rule-based and ML systems is not always straightforward and often depends on the specific language, task requirements, and available resources. For Bengali text normalization:

* **Complexity vs. Need:** While ML models offer powerful sequence transduction capabilities, developing, training, and deploying a state-of-the-art neural model specifically for the diverse requirements of Bengali normalization is a substantial undertaking. It requires large datasets, significant computational resources, and expertise in ML model tuning.
* **The “Good Enough” Principle and Simplicity:** Our rule-based system demonstrates that a very high level of accuracy and coverage for the most frequent and critical NSW categories in Bengali can be achieved through direct, interpretable rules. It provides a robust, understandable, and resource-efficient solution that meets the needs of many practical applications *today*. This exemplifies a scenario where a simpler, more direct approach grounded in linguistic knowledge can be more advantageous than immediately resorting to complex, data-hungry ML models. It tackles the core problem effectively without the associated overhead.
* **Hybrid Potential:** The ideal future might lie in hybrid systems [Placeholder Reference]. Rules could handle high-confidence, well-defined patterns (like standard dates, times, currency), while ML models could act as a fallback for ambiguous cases or highly variable/informal text. The output of the rule-based system could even be used to generate large amounts of synthetic data to bootstrap the training of ML models.

**5.4. Significance for Bengali Language Technologies**

The availability of a comprehensive text normalizer, like the one described, is a foundational step for advancing Bengali NLP and speech technology. By providing clean, consistent, and optionally phonetic input, it can directly lead to:

* More natural and accurate Bengali TTS systems.
* Improved recognition rates for Bengali ASR systems by reducing vocabulary variations.
* More reliable performance in downstream NLP tasks like machine translation, information retrieval, and text analysis that depend on consistent entity representation.
* Lowering the barrier for researchers and developers working on Bengali language applications by providing a crucial preprocessing component.

**6. Conclusion and Future Work**

This paper presented a comprehensive rule-based system for Bengali text normalization, designed to address the specific linguistic and orthographic challenges of the language. Through a modular pipeline architecture combining regular expression-based pattern identification, procedural conversion logic, and curated linguistic resources, the system effectively handles a wide range of non-standard word categories, including numbers with Bengali scaling units, complex ordinals, diverse date/time formats, currency, percentages, and code-mixed English terms via phonetic transliteration. An integrated Bengali-to-IPA conversion module further enhances its utility for speech applications.

We argued that despite the prevalence of machine learning in NLP, a well-designed, linguistically informed rule-based approach remains highly relevant and effective for Bengali text normalization. Its strengths in interpretability, precision, control over linguistic rules, and data efficiency offer significant advantages, providing a pragmatic and robust solution in the current resource context. The system serves as a practical tool for preprocessing Bengali text and a strong baseline for future research.

Future work should focus on several key areas:

* **Expanding Coverage and Robustness:** Continuously refining rules to handle more linguistic variations, informal constructs (e.g., slang, social media abbreviations), and potential ambiguities. Incorporating more sophisticated context analysis might be necessary.
* **Enriching Linguistic Resources:** Systematically expanding the English-to-Bengali phonetic map and the IPA conjunct dictionary to cover a broader vocabulary and rarer consonant clusters.
* **Developing Standardized Benchmarks:** The most critical need is the creation and community adoption of large-scale, publicly available, annotated Bengali text normalization corpora. This will enable rigorous quantitative evaluation and objective comparison between different approaches (rule-based, ML, hybrid).
* **Exploring Hybrid Models:** Investigating architectures that combine the strengths of this rule-based system with machine learning components. Rules could handle deterministic cases, while ML models could manage ambiguity or unseen patterns, potentially trained using data generated or augmented by the rule-based system.
* **Addressing Dialectal Variation:** Investigating the extent of normalization differences required for major Bengali dialects (e.g., Standard Colloquial Bengali vs. regional variations) and incorporating dialect-specific rules if necessary.

The codebase for the described system is available for review and further development at: [[**HERE**](https://github.com/shohanur-shoron/Bangla-Cleaner.git)].

**References**

[1] Sproat, R., et al. (2001). Normalization of non-standard words. *Computer Speech & Language*, 15(3), 287-333.

[2] Taylor, P. (2009). *Text-to-Speech Synthesis*. Cambridge University Press.

[3] Allen, J., Hunnicutt, M. S., & Klatt, D. (1987). *From text to speech: The MITalk system*. Cambridge University Press. [Placeholder Reference - Classic Rule-Based TTS/Norm]

[4] Kaplan, R. M., & Kay, M. (1994). Regular models of phonological rule systems. *Computational linguistics*, 20(3), 331-378.

[5] Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2), 269-311.

[6] Goyal, V., & Lehal, G. S. (2008). Text normalization in Hindi. *In Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*. [Placeholder Reference]

[7] Ramanarayanan, V., et al. (2011). Text normalization for Malayalam text-to-speech synthesis system. *International Journal of Computer Applications*, 19(7), 1-6. [Placeholder Reference]

[8] Zhang, Y., et al. (2019). End-to-End Text Normalization with Neural Sequence-to-Sequence Models. *arXiv preprint arXiv:1904.08017*.

[9] Lusetti, M., et al. (2021). Transformer-based Text Normalization. *arXiv preprint arXiv:2110.07150*.

[10] Ebden, P. & Sproat, R. (2015). The Kestrel TTS text normalization system. *Natural Language Engineering*, 21(3), 333–353. [Discusses rule-based advantages/design]

[11] Behera, L., et al. (2010). Text normalization for Odia language. *In Proceedings of the 1st International Conference on Intelligent Interactive Technologies and Multimedia*. [Placeholder Reference - Indic Example]

[12] Dave, S., et al. (2003). Text processing for text-to-speech systems: Handling normalization and ambiguity. *Speech Communication*, 41(2-3), 321-333. [Placeholder Reference - General Norm Issues]

[13] Gorman, K., & Sproat, R. (2016). Minimally supervised models for grammar-based text normalization. *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [Placeholder Reference - Hybrid/ML]

[14] Vaněk, J., et al. (2021). Rule-based vs. Neural Text Normalization for Czech Speech Synthesis. *In Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC 2020)*. [Placeholder Reference - Rule vs Neural comparison for another lang]

[15] Das, A., & Bandyopadhyay, S. (2009). Handling numerals and dates in Bengali text processing. *In Proceedings of the 11th International Conference on Information Technology (ICIT-2009)*. [Placeholder Reference - Specific Bengali Numerals/Dates]

[16] Ekbal, A., et al. (2008). Bengali Named Entity Recognition using Support Vector Machine. *In Proceedings of the 6th International Conference on Natural Language Processing (ICON-2008)*. [Placeholder Reference - Related Bengali NLP showing complexity]

[17] Chakravarthi, B. R., et al. (2020). A dataset for text normalization of code-mixed social media text in Tamil, Telugu, and Malayalam. *In Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*. [Placeholder Reference - Code-Mixing Data/Norm]