A Study of Reduplicated Forms for Bengali Text Processing

Thesis submitted by
Abhijit Barman

Doctor of Philosophy (Engineering)

Department of Computer Science and Engineering
Faculty Council of Engineering & Technology
Jadavpur University
Kolkata, India

2025

JADAVPUR UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY

INDEX NO. **130/19/E**

1. **Title of the Thesis**: A Study of Reduplicated Forms for Bengali Text Processing

2. **Name, Designation and Institution of the Supervisor/s:**

   (a) **Prof. Diganta Saha**
   Professor, Department of Computer Science and Engineering
   Jadavpur University, Kolkata –700032

   (b) **Dr. Alok Ranjan Pal**
   Assistant Professor, Department of Computer Science and Engineering
   College of Engineering and Management, Kolaghat –721171

3. **List of Publications:**

   (a) **Journals**

      i. A. Barman, D. Saha, and A. R. Pal, "Bengali reduplication generation with finite-state transducers (FSTs)", International Journal of Speech Technology, 2024, , vol. 27, Issue 03, pp. 729-737, doi: 10.1007/s10772-024-10124-6.

      ii. A. Barman, D. Saha, and A. R. Pal, "An Upgraded Approach for Identifying Partially Reduplicated Forms in Bengali Text", SN Computer Science, 2024, vol. 05, Issue 07, pp. 892, doi: 10.1007/s42979-024-03069-9

      iii. A. Barman, D. Saha, and A. R. Pal, "Identification and Sense Tagging of Reduplicated Forms in Bengali", Transactions on Asian and Low-Resource Language Information Processing, 2025 (Minor Revision Submitted)

   (b) **International Conferences**

      i. A. Barman, and D. Saha, "Algorithm for Removal of Semantically Insignificant Content Words", International Journal of Computer Sciences and Engineering, 2019, vol. 07, Issue. 01, pp. 53-56.

      ii. A. Barman, D. Saha and A. R. Pal, "An Approach for Maintaining Structural Uniformity of Multiword Expressions", 2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2024, pp. 1-4, doi: 10.1109/ICAECT60202.2024.10469553.

4. **List of Patents:** None

5. **List of Presentations in National/International Conferences and Workshops:** None

JADAVPUR UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY

STATEMENT OF ORIGINALITY

I, **Abhijit Barman** registered on **13th June, 2019**, do hereby declare that this thesis entitled **"A Study of Reduplicated Forms for Bengali Text Processing"** contains literature survey and original research work done by the undersigned candidate as part of Doctoral studies.

All information in this thesis have been obtained and presented in accordance with existing academic rules and ethical conduct. I declare that, as required by these rules and conduct, I have fully cited and referred all materials and results that are not original to this work.

I also declare that I have checked this thesis as per the "Policy on Anti Plagiarism, Jadavpur University, 2019", and the level of similarity as checked by iThenticate software is 8 %.

Signature of the Candidate :

Abhijit Barman.

Date : 30/04/2025

Certified by Supervisors :
(Signature with date, seal)

1. Diganta Saha 30/4/25

(Prof. Diganta Saha)

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata - 700 032

2. Alok Ranjan Pal 30/4/25

(Dr. Alok Ranjan Pal)

Assistant Professor
Dept. of Computer Sc. & Engg
College of Engg. & Mgmt., Kolaghat
PIN - 721171, WB

JADAVPUR UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE FROM THE SUPERVISOR/S

This is to certify that the thesis entitled **"A Study of Reduplicated Forms for Bengali Text Processing"** submitted by **Shri Abhijit Barman**, who got his name registered on **13th June, 2019** for the award of **Ph.D. (Engg.)** degree of Jadavpur University is absolutely based upon his own work under the supervision of **Prof. Diganta Saha** and **Dr. Alok Ranjan Pal** and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

1. Diganta Saha 30/4/25

(Prof. Diganta Saha)

Signature of the Supervisor
and date with Official seal

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata - 700 032

2. Alok Ranjan Pal 30/4/25

(Dr. Alok Ranjan Pal)

Signature of the Supervisor
and date with Official seal

Assistant Professor
Dept. of Computer Sc. & Engg
College of Engg. & Mgmt., Kolaghat
PIN - 721171, WB

Acknowledgements

I express my deepest gratitude to my doctoral research advisors, Professor Diganta Saha and Dr. Alok Ranjan Pal, for their unwavering support and inspiration throughout my journey. They allowed me to pursue the problems I was passionate about and always took the time to listen to my ideas, even when some were less promising. Their valuable suggestions helped refine my thoughts, and their own insights greatly enriched my work. This thesis would not have been possible without their collaboration and support.

I am deeply grateful to Professor Niladri Sekhar Dash for his insightful vision and generous guidance in identifying the scope of this research. I extend my special thanks to him for his invaluable advice and kind suggestions at every stage of this work.

I would like to extend my sincere thanks to my co-researcher, Ratul Das. Without his assistance, many of the results in my thesis might not have materialized. I greatly appreciate the valuable discussions and ideas we've exchanged over the past few years.

I express my heartfelt gratitude to my parents and teachers for their inspiration and encouragement to pursue this PhD. I am also deeply thankful to my wife and our son for their patience and continuous support throughout the journey.

Abhijit Barman
PhD Fellow, Jadavpur University

Abstract

Reduplication is a prominent linguistic phenomenon in Bengali, serving various grammatical and semantic functions such as intensification, plurality, and emphasis. Its complex nature, characterized by structural variations and diverse usages, poses significant challenges for computational processing. This research addresses these challenges through a multi-phase approach aimed at enhancing the accuracy of identifying, generating, and sense-tagging reduplicated forms, which are critical for applications such as machine translation and sentiment analysis.

This research introduces a precise rule-based methodology for detecting partially reduplicated Multi-Word Expressions (MWEs) in Bengali texts. The process is executed in two phases to enhance accuracy and effectiveness. In the first phase, a Levenshtein distance based algorithm identifies partially reduplicated forms by assessing word similarity and flagging relevant instances. The second phase refines this output using a noble technique known as *Word Expansion*. Word Expansion technique, significantly improving detection performance, as reflected by metrics of 90.00% Precision, 85.71% Recall, and an F1-Score of 87.80%.

Additionally, this study leverages supervised machine learning to identify reduplicated MWEs in Bengali corpora. Candidate MWEs are extracted using syntactic rules, followed by classification through a Random Forest algorithm. This approach employs diverse features, such as association measures and linguistic attributes, achieving superior performance with 90.90% Precision, 95.24% Recall, and an F1-Score of 93.02%. These findings highlight the potential of both rule-based and machine learning methodologies in addressing the structural diversity and complexity of reduplication in Bengali texts.

Further, the study models the generation of complete and partial reduplications using two-way finite-state transducers (FSTs). While complete reduplications are effectively modeled with a single FST, partial reduplications require multiple FSTs to accommodate their varied patterns. This process achieves an F1-Score of 88.11%, demonstrating its utility in identifying reduplication instances.

Finally, a robust sense-tagging framework for reduplicated forms leverages BERT-based word embeddings. By clustering word vectors based on cosine similarity scores, the system achieves distinct sense identification, with F1-Scores of 84.96%. These findings underscore the potential of advanced NLP techniques to effectively process and understand the complexities of Bengali reduplicated forms.

This research provides valuable insights into computational linguistics, offering scalable solutions for NLP tasks in low-resource language scenarios.

Contents

List of Figures

List of Tables

List of Abbreviations

A

AI	**A**rtificial **I**ntelligence
ALBERT	**A** **L**ight **BERT**

B

BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformer
BiLSTM	**Bi**directional **L**ong **S**hort-**T**erm **M**emory

C

CNN	**C**onvolutional **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
CRF	**C**onditional **R**andom **F**ield

D

DB	**D**ata **B**ase
DCT	**D**iscrete **C**osine **T**ransform
DFA	**D**eterministic **F**inite **A**utomata

E

ELMo	**E**mbeddings from **L**anguage **M**odel

F

FBI	**F**uzzy **B**igram **I**ndex
FN	**F**alse **N**egatives
FP	**F**alse **P**ositives
FSM	**F**inite **S**tate textbfMachine
FST	**F**inite **S**tate **T**ransducers

G

GCN	**G**raph **C**onvolutional **N**etwork
GloVe	**G**lobal **V**ectors for **W**ord R**e**presentation
GPT	**G**enerative **P**re-trained **T**ransformer

H

HMM	**H**idden **M**arkov **M**odel

I

ICF	**I**nverse **C**lass **F**requency
IE	**I**nformation **E**xtraction
IR	**I**nformation **R**etrieval

L

LSTM	**L**ong **S**hort-**T**erm **M**emory

M

MDT	**M**orphological **D**oubling **T**heory
MeitY	**M**inistry of **E**lectronics and **I**nformation **T**echnology
MLM	**M**asked **L**anguage **M**odeling
MWE	**M**ulti-**W**ord **E**xpression

N

NLP	**N**atural **L**anguage **P**rocessing
NSP	**N**ext **S**entence **P**rediction

P

PMI	**P**ointwise **M**utual **I**nformation
POS	**P**arts-**O**f-**S**peech

R

RNN	**R**ecurrent **N**eural **N**etwork
RoBERTa	**Ro**bustly **O**ptimized **BERT a**pproach

S

SVM	**S**upport **V**ector **M**achine
T	
TDIL	**T**echnology **D**evelopment for **I**ndian **L**anguages
TF-IDF	**T**erm **F**requency-**I**nverse **D**ocument **F**requency
TN	**T**rue **N**egatives
TP	**T**rue **P**ositives

List of Symbols

A list of all the symbols, their representative quantities and units has been provided below so as to familiarize the readers with the frequently used symbols within this thesis.

Symbols for Physical Systems

δ	State transition function
Γ	Bengali alphabet with word separator(-)
κ	Corpus
ω	Weight measure
Σ	Bengali alphabet
C	Class or Category
c	Character
D	Term-Document Matrix
d	Document
F	Set of accepting states
H	Hypothesis
H_0	Null Hypothesis
L_d	Levenshtein distance
M	2-way deterministic FST
P	Probability
Q	Finite set of states
q	State variable
q_0	Initial state
T_h	Threshold value

w Word

$<w^1 w^2>$ Bi-gram

s,t Strings

CHAPTER 1

Introduction

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) focused on enabling machines to comprehend, interpret, and generate human (natural) language. This capability allows machines to interact with humans in a natural and intuitive manner. The process of machine interaction with human language can be divided into several key steps. The first step in NLP is recognizing the human language. This involves identifying the language being used (e.g., English, Bengali) and configuring the system to process that specific language. Once the language is recognized, the machine must understand it by parsing the input to derive meaning, which involves several stages of analysis. After understanding the input, the machine retrieves relevant information from its database or external sources, crucial for answering questions or providing recommendations. The retrieved information is organized, logical reasoning is applied, and the information is prepared for response generation. Inference involves deriving conclusions from the processed information, such as predicting outcomes or making decisions based on the data. Finally, the machine generates a natural language response, constructing sentences that are grammatically correct and contextually appropriate. To process human language (spoken or written), the machine first splits the input into sentences using termination symbols like periods, exclamation marks, or question marks. Each sentence is then broken into individual words or tokens. This step is crucial for subsequent processing stages. Morphological analysis involves identifying and separating root forms of words from their inflections, aiding in understanding their base meanings. Syntax analysis, or parsing, deals with the structural arrangement of words in a sentence, identifying grammatical structures to understand syntactic relationships. Lexical semantics focuses on the meaning of individual words, looking up meanings, synonyms, and antonyms in a dictionary or lexicon. Compositional semantics deals with meanings arising from word combinations, considering how individual words contribute to the sentence's overall

meaning. Pragmatics involves understanding language in context, using appropriate words and phrases according to the situation and social context to achieve conversational goals. Discourse analysis extends beyond sentences to understand language in larger contexts, examining coherence, cohesion, and sentence relationships to derive overall meaning. NLP has various applications, including spelling correction, grammar checking, machine translation, information extraction, sentiment analysis, speech recognition, and text summarization. However, NLP also faces challenges such as ambiguity, where words and sentences can have multiple meanings; contextual understanding, which is crucial for accurate interpretation; cultural nuances, requiring systems to account for differences in language use across cultures; and idiomatic expressions that cannot be interpreted literally. By addressing these challenges and leveraging advanced techniques such as machine learning and deep learning, NLP aims to create systems that can interact with humans in a more natural and effective way, making technology more accessible and useful in everyday life.

1.1 What is Multi-Word Expression (MWE)?

In natural languages, groups of words can form a cohesive semantic unit known as Multi-Word Expressions (MWEs). The semantics of an MWE cannot be deduced by simply combining the meanings of its individual words. Take the Bengali idiom বালির বাঁধ (*bālir bāndh*: fragile), for instance, whose literal word-by-word translation into English is 'of sand dam.' However, when considered as a whole, বালির বাঁধ (*bālir bāndh*) translates to 'fragile' in English. Due to their idiomatic nature, MWEs present a significant challenge in Computational Linguistics. This challenge is more substantial than anticipated, given that the usage of MWEs in literature is comparable to that of single words. For instance, in English WordNet 1.7, approximately 41% of the entries consist of MWEs [1]. MWEs find diverse applications in the domain of Natural Language Processing (NLP) and hold theoretical significance.

Multi-Word Expressions (MWEs) are a fascinating and complex aspect of natural language that pose significant challenges in both linguistic analysis and computational processing. MWEs encompass a variety of types, each with unique characteristics that make them distinct. Understanding these types is essential for effective natural language processing (NLP), as MWEs often carry meanings that cannot be deduced from their individual components.

1.2 What is Reduplication?

Reduplication, within the broader scope of linguistic description, is a phenomenon characterized by the repetition or duplication of linguistic elements, ranging from individual phonemes and morphemes to entire words, phrases, clauses, or utterances. This process represents a robust morphological mechanism where the root or stem of a word, or a segment of it, is duplicated with precision or undergoes slight alterations. Reduplication serves as a versatile and effective word formation strategy, employed in both inflections and lexical derivations.

In the context of inflections, reduplication plays a crucial role in conveying various grammatical functions, such as indicating plurality, intensification, continuation, and more. It enhances the expressive quality of language, allowing speakers to adopt tones that go beyond the ordinary and inject additional meaning into their expressions. This makes reduplication a valuable tool for capturing nuanced shades of meaning and conveying a more vivid or figurative sense.

Furthermore, reduplication is not confined to a single linguistic function; it is employed in lexical derivation to create new words. This demonstrates its versatility in contributing to the expansion and enrichment of a language's vocabulary.

The richness and robustness of the reduplication process in Bengali are noteworthy, allowing virtually every word in the language to undergo reduplication. However, orthographic inconsistencies often lead to reduplicated words being written with or without a space between the constituent parts. The lexical element subject to reduplication can take various forms, including a word, morpheme, syllable, sequence of syllables, or a string of characters that doesn't form a prosodic constituent like a morpheme, syllable, or root. It is noteworthy that reduplication is a widespread phenomenon found in languages across the globe, although its level of linguistic productivity can vary significantly from one language to another. Despite these variations, reduplication stands as a dynamic and expressive linguistic feature with applications in both structural and creative aspects of language use.

Observations reveal that reduplication extends across all word classes, encompassing pronouns and indeclinables. In many instances, the process conveys the idea of the plurality of an item or the continuity of an action. Additionally, the realm of reduplication includes various echoic and onomatopoeic words prevalent in the language, adding to its linguistic diversity.

1.3 Types of Reduplication

In every language, the duplication of forms can be categorized into two overarching types: (a) repetition occurring at the level of expression or form, and (b)

repetition occurring at the level of content or semantics.

(a) **Syntactic Reduplication:** This repetition occurs specifically at the level of expression or form, meaning that the structure or syntax of the language is used to duplicate words, phrases, or even entire clauses. It is classified into two distinct categories.

i. **Complete Reduplication:** Here an entire word or phrase is repeated exactly, without any change in its form. It is further divided into three types.

- **Lexical Reduplication:** Lexical reduplication is a linguistic phenomenon characterized by the repetition of a sequence of phonological units within a word or one of its components. This process involves duplicating either an entire word, a word stem (consisting of the root along with one or more affixes), or the root itself to generate reduplicated forms. Importantly, both the original and duplicated segments of the resultant form operate as independent words, each maintaining its unique lexical identity within the language. These individual words possess distinct meanings when considered separately. However, when they are repetitively combined, the resulting meaning is not simply a summation of the individual meanings but an expression with unique semantic connotations.

হাজার হাজার (*hājār hājār*: thousands)
বড় বড় (*baṛa baṛa*: large)
ভাল ভাল (*bhāla bhāla*: of good qualities)
দিন দিন (*din din*: day by day)
সকাল সকাল (*sakāl-sakāl*: in early morning)
ঝাল ঝাল (*jhāl jhāl*: spicy)

These examples illustrate a phenomenon where the second part of each reduplicated word maintains its autonomy as a complete and standalone word, possessing its distinct lexical identity and meaning. However, in the reduplicated form, it creates a new meaning that differs from the literal interpretation of its individual parts. In certain instances, this repetition gives rise to an idiomatic sense, characterized by a unique meaning that cannot be directly inferred from the individual components of the reduplicated word. The process of reduplication, therefore, offers a creative and context-specific mechanism for semantic expression in Bengali.

- **Inflected Reduplication:** This particular type of reduplication exhibits a unique characteristic wherein the initial word undergoes inflection, and the resulting inflected form is duplicated to produce the final reduplicated structure. Importantly, these inflected forms, while capable of standing independently as words within a text, possess the freedom to be repetitively used, giving rise to reduplicated forms. Notably, this repetition occurs without introducing any structural changes or alterations to the inflected forms during the reduplication process. The flexibility of these inflected forms to seamlessly contribute to the generation of reduplicated expressions adds a distinctive aspect to this specific reduplication phenomenon.

  দেখে দেখে (*dekhe dekhe*: seeing)
  ঘরে ঘরে (*ghare ghare*: in every house)
  মুখে মুখে (*mukhe mukhe*: verbally)
  কাজে কাজে (*kāe kāje*: in all works)
  পথে পথে (*pathe pathe*: in all roads)
  মনে মনে (*mane mane*: in mind or silently)
  হাতে হাত (*hāte hāt*: directly)
  বলতে বলতে (*balte balte*: while saying)

  These are some instances of inflected reduplication. From a semantic perspective, these reduplicated forms convey a diverse range of meanings, encompassing concepts such as plurality, repletion, recurrence, proximity, continuation, indefinite time and location, and various facets associated with events and actions. The richness of these reduplicated expressions lies in their ability to capture a spectrum of senses, making them a versatile linguistic tool for communicating different shades of information and emphasizing various aspects of events or actions within the context of the language.

- **Onomatopoeic reduplication:** The incorporation of reduplicated words to mimic sounds is a widespread phenomenon in the Bengali language. This linguistic practice involves the repetition of words to imitate natural sounds, and it has been explicitly stated that when reduplicated words are used in an onomatopoeic context, they exhibit a diverse range of formations and meanings. According to a recent study, Bengali is home to around 2,500 onomatopoeic words, with a substantial portion (approximately 50%) of this lexicon comprising

reduplicated forms [2]. This suggests a significant reliance on reduplication as a linguistic mechanism to capture and convey various auditory experiences in the language.

ঝনঝন (*jhanjhan*: sound of breaking glasses)
খটখট (*khatkhat*: sound of knocking on doors)
টুপুর টুপুর (*tupur tāpur*: sound of rain)
খলখল (*khalkhal*: sound of laughter)
ধুপধুপ (*dhupdhup*: sound of falling)

These reduplicated words play a versatile role in linguistic expressions, particularly in representing diverse auditory experiences. In the context of animal and bird sounds, as well as noises emanating from objects, natural environments, tools, and machines, these reduplicated forms effectively capture the essence of these acoustic phenomena. Furthermore, they extend their usage to human actions and performances, encapsulating a range of activities such as laughter, crying, coughing, dancing, drinking, eating, walking, speaking, sleeping, running, and cutting. This broad application highlights the flexibility and richness of these reduplicated terms in conveying a wide spectrum of sounds and human behaviors within the linguistic framework.

ii. **Partial Reduplication:**

- **Morphological Reduplication:** Morphological reduplication is a linguistic process involving the repetition of morphemes, the smallest meaning-bearing units in a language. These repeated morphemes often consist of iterated syllables and contribute to forming words with distinct meanings. It is commonly observed in onomatopoeic words, sound symbolism, mimicry, and echo words. This process enhances language expressiveness by imitating sounds, symbolizing actions, or representing specific qualities through repetition. Morphological reduplication spans various linguistic categories, enriching vocabulary with symbolic and descriptive dimensions.

  The Morphological Doubling Theory (MDT) is an approach to reduplication that requires two instances of the same morphological element, defined by meaning rather than sound [3]. This doubling can

apply to an entire word, stem, root, or affix. Unlike phonological processes, MDT is driven by morphological requirements, with identity based on morphosemantic content, not phonological similarity.

The process includes variations such as consonant gemination, vowel lengthening, or near-exact replication of the base. In Bengali, a common form involves modifying the first letter or initial syllable of the first word to create the second, ensuring a phonological match. This alignment harmonizes the ending sounds of both words, establishing phonological correspondence while retaining the meaning of the initial word.

বই টই (*bai-tai*: books and other things)
মাছ টাছ (*māch-tāch*: fish and other things)
সময় টময় (*samay-tamay*: time and other items)
হাতে নাতে (*hāte-nāte*: at red hand)
বাড়ি টাড়ি (*bāṛi-tāṛi*: houses and other things)
জল টল (*jal-tal*: water and other things)
ছেলে টেলে (*chele-tele*: boys and others)
খেলা টেলা (*khelā-telā*: games and other actions)

The final words in these examples can be broken down to identify the meaningless sequences that make up the second part, which are then used in the formation of reduplicated words. The repeated segment is not a lexical item or part of one, and it only gains meaning after the process of reduplication. Additionally, unlike fully reduplicated words, which usually convey a distributive meaning, partially reduplicated words in Bengali often suggest an "Et Cetera" or "X and alike" interpretation.

- **Correlative Reduplication:** Correlative reduplication is a linguistic phenomenon wherein the second word in a pair is formed from the initial word, typically a verb. This process involves specific phonological alterations, such as the transformation from ā to i, ultimately resulting in the creation of the reduplicated form. The key characteristic of correlative reduplication is the alignment of morphophonological structures between the original and derived words. This linguistic mechanism allows for a relationship between the paired words, often serving specific communicative or expressive purposes

within the language.
কাটাকাটি (*kātākāti*: cutting)
টানাটানি (*tānātāni*: pulling)
ঠেলাঠেলি (*thelātheli*: pushing)
বলাবলি (*balābali*: saying)
দেখাদেখি (*dekhadekhi*: seeing)
হাতাহাতি (*hātāhāti*: fighting with hands)
হানাহানি (*hānāhāni*: killing)
কোলাকুলি (*kola-kuli*: embracing)
খোলাখুলি (*khola-khuli*: openly)

Semantically, these forms serve to express the idea of exchange, barter, or interchange of actions, as exemplified by the provided examples.

(b) **Semantic Reduplication:** Semantic reduplication in Bengali predominantly manifests in forms where the paired words exhibit semantic relations. In these instances, both the initial and repeated elements within the pair convey nearly identical or closely related meanings, with instances of rare occurrences where opposite senses may be represented. The subsequent examples illustrate this phenomenon:
বিয়ে সাদি (*biye sādi*: marriage and weddings)
টাকা পয়সা (*tākā paysā*: money and penny)
বই পত্র (*bai-patra*: books and papers)
রাজা প্রজা (*rājā-prajā*: king and subject)
লোক জন (*lok-jan*: people and public)
মাথা মুন্ডু (*māthā-muṇḍu*: head and head)
কথা বার্তা (*kathā bārtā*: words and talks)
ছোট খাটো (*chota-khāto*: little and small)
দিন রাত (*din rāt*: day and night)
সকাল বিকাল (*sakāl bikāl*: morning and afternoon)

These pairs of words can be regarded as instances of synonyms, hyponyms, or antonyms, indicating some form of semantic relatedness between their members. However, it is important to note that mere semantic relatedness is not enough to categorize them as reduplicated forms. Therefore, it is more accurate to classify them as examples of copulative compounds[4] rather than reduplicated forms, as they lack formal properties or elements typically associated with reduplication.

1.4 Structures of Bengali Reduplication

Regarding the formation of reduplication across different parts of speech in Bengali, it is notably productive and frequent for nouns, adjectives, finite and non-finite verbs, adverbs, and postpositions. However, its occurrence is moderate in the case of indeclinables and pronouns.

(a) **Reduplication of Nouns:** Structural analysis of Bengali nouns reveals six patterns of reduplication:

i. The base form is duplicated without any suffix or case marker: ঘর ঘর (*ghar ghar*: evey house), দিন দিন (*din din*: regular), মেয়ে মেয়ে (*meye meye*: like a girl), টাকা টাকা (*tākā tākā*: money), জল জল (*jal jal*: water) etc.

ii. The base form is duplicated, with the second member as an echo form: ফল টল (*phal tal*: fruits, etc.), হাত টাত (*hāt tāt*: hand, etc.), বই টই (*bai tai*: books etc.).

iii. The base form is duplicated with the suffix *-e* on both members: জনে জনে (*jane jane*: person by person), দিনে দিনে (*dine dine*: day by day), ঘরে ঘরে (*ghare ghare*: in every house).

iv. The base form is duplicated with the suffix *-āy* on both members: কথায় কথায় (*kathāy kathāy*: in every word), দরজায় দরজায় (*darjāy darjāy*: at every door), রাস্তায় রাস্তায় (*rāstāy rāstāy*: on different roads).

v. The base form is duplicated with the suffix *-te* on both members: নদীতে নদীতে (*nadīte nadīte*: in rivers), বাড়িতে বাড়িতে (*bāṛite bāṛite*: in every house), সখীতে সখীতে (*sakhīte sakhīte*: between two girlfriends).

(b) **Reduplication of Pronouns:** The pronouns in Bengali are reduplicated in the following ways:

i. Pronominal roots are reduplicated without any suffix or inflection marker: যে যে (*ye ye*: those who), যা যা (*yā yā*: that which), সে সে (*se se*: those), তুমি তুমি (*tumi tumi*: you and you), যে সে (*ye se*: anyone), কে কে (*ke ke*: who and who), কি কি (*ki ki*: which ones), কেউ কেউ (*keu keu*: someone), কিছু কিছু (*kichu kichu*: some), কারো কারো (*kāro kāro*: someone), নিজে নিজে (*nije nije*: by self) etc.

ii. Pronominal roots are reduplicated with the addition of the enclitic marker *-tā* or *-ti:* এটা এটা (*etā etā*: this and this), এটা সেটা (*etā setā*: this and that), এটা সেটা (*etā setā*: anything), এটি এটি (*eti eti*: anything), সেটা

সেটা (*setā setā*: that and that), কোনটা কোনটা (*kontā kontā*: which ones), কোনটি কোনটি (*konti konti*: which ones) etc.

iii. Pronominal roots are reduplicated with the plural marker *-rā:* যারা যারা (*yarā yārā*: those who), যারা তারা (*yarā tārā*: one and all), কারা কারা (*kārā kārā*: who ones) etc.

iv. Pronominal roots are reduplicated with the plural marker *-gulo:* এগুলো এগুলো (*egulo egulo*: those which), এগুলো সেগুলো (*egulo segulo*: one and all), কোনগুলো কোনগুলো (*kongulo kongulo*: which ones) etc.

v. Pronominal roots are reduplicated with the accusative case marker *-ke:* যাকে যাকে (*yāke yāke*: to whom), যাকে তাকে (*yāke tāke*: to anyone), একে একে (*eke eke*: to this and this), ওকে ওকে (*oke oke*: to that and that), তাকে তাকে (*tāke tāke*: to him and him), একে তাকে (*eke tāke*: to this and him), একে ওকে (*eke oke*: to this and that), কাকে কাকে (*kāke kāke*: to whom and whom), কাউকে কাউকে (*kāuke kāuke*: to someone) etc.

vi. Pronominal roots are reduplicated with the genitive case marker *-r:* যার যার (*yār yār*: of whom), যার তার (*yār tār*: of anyone), কার কার (*kār kār*: of whom), and নিজের নিজের (*nijer nijer*: of own) etc.

vii. Pronominal roots are reduplicated with the genitive case marker *-der:* যাদের যাদের (*yāder yāder*: of whom), যাদের তাদের (*yāder tāder*: of anyone), and কাদের কাদের (*kāder kāder*: of whom) etc.

(c) **Reduplication of Finite Verbs:** The finite verbs in Bengali are reduplicated in the following ways:

i. The root is duplicated without any suffix or marker, e.g., চল চল (*cal cal*: walk), ধর ধর (*dhar dhar*: catch), বল বল (*bal bal*: say), দেখ দেখ (*dekh dekh*: see), and মর মর (*mar mar*: die) etc.

ii. The root is duplicated with the third person present tense marker *-e*, e.g., করে করে (*kare kare*: does), বলে বলে (*bale bale*: says), চলে চলে (*cale cale*: runs), and ধরে ধরে (*dhare dhare*: catches) etc.

iii. The root is duplicated with the third person past tense marker *-la*, e.g., করল করল (*karla karla*: did), শুনল শুনল (*sunla sunla*: heard), দেখল দেখল (*dekhla dekhla*: saw), and বলল বলল (*balla balla*: said) etc.

iv. The root is duplicated with the past tense marker *-le*, e.g., করলে করলে (*karle karle*: did), শুনলে শুনলে (*sunle sunle*: heard), দেখলে দেখলে (*dekhle dekhle*: saw), and বললে বললে (*balle balle*: said) etc.

v. The root is duplicated with the second person past tense marker *-ile*, e.g., করিলে করিলে (*karile karile*: did), শুনিলে শুনিলে (*sunile sunile*: heard), দেখিলে দেখিলে (*dekhile dekhile*: saw), and বলিলে বলিলে (*balile balile*: said) etc.

vi. The root is duplicated with the first person future tense marker *-ba*, e.g., শুনবো শুনবো (*sunbo sunbo*: will hear), দেখবো দেখবো (*dekhbo dekhbo*: will see), and ভাববো ভাববো (*bhābbo bhābbo*: will think) etc.

vii. The root is duplicated with the second/third person future tense marker *-be*, e.g., শুনবে শুনবে (*sunbe sunbe*: will hear), দেখবে দেখবে (*dekhbe dekhbe*: will see), and বলবে বলবে (*balbe balbe*: will say) etc.

viii. The first word is a verb root while the second word is an echo form of the first word, e.g., খায় দায় (*khāy dāy*: eats and does other things), আসে টাসে (*āse tāse*: comes rarely), শোনে টোনে (*sone tone*: listens almost carelessly), and দেখে টেখে (*dekhe tekhe*: sees casually) etc.

(d) **Reduplication of Non-Finite Verbs:** Reduplication of Non-Finite Verbs in Bengali occurs as follows:

i. The root is duplicated with the marker *-e*, e.g., বলে বলে (*bale bale*: saying), চলে চলে (*cale cale*: walking), ধরে ধরে (*dhare dhare*: catching), and দেখে দেখে (*dekhe dekhe*: seeing) etc.

ii. The root is duplicated with the marker *-te*, e.g., শুনতে শুনতে (*sunte sunte*: hearing), দেখতে দেখতে (*dekhte dekhte*: seeing), ভাবতে ভাবতে (*bhābte bhābte*: thinking), and চলতে চলতে (*calte calte*: walking) etc.

iii. The root is duplicated with the marker *-iyā*, e.g., শুনিয়া শুনিয়া (*suniyā suniyā*: hearing), দেখিয়া দেখিয়া (*dekiyā dekhiyā*: seeing), বলিয়া বলিয়া (*baliyā baliyā*: saying), and চলিয়া চলিয়া (*caliyā caliyā*: walking) etc.

iv. The root is duplicated with the marker *-ite*, e.g., শুনিতে শুনিতে (*sunite sunite*: hearing), দেখতে দেখতে (*dekhte dekhte*: seeing), ভাবিতে ভাবিতে (*bhābite bhābite*: thinking), and চলিতে চলিতে (*calite calite*: walking) etc.

v. The root is duplicated with the marker *-iye*, e.g., দেখিয়ে দেখিয়ে (*dekhiye dekhiye*: showing to others), শুনিয়ে শুনিয়ে (*suniye suniye*: making others hear), খেলিয়ে খেলিয়ে (*kheliye kheliye*: making others play), and হাসিয়ে হাসিয়ে (*hāsiye hāsiye*: making others laugh) is duplicated with the marker *-le*, e.g., করলে করলে (*karle karle*: having done), শুনলে শুনলে (*sunle sunle*: having heard), দেখলে দেখলে (*dekhle dekhle*: having seen), and বললে বললে (*balle balle*: having said) etc.

vi. The first word is a verb root inflected with the marker -*e*, while the second word is an echo of the first word using the same marker, e.g., খেয়ে দেয়ে (*kheye deye*: eating and doing other things), লুটে পুটে (*lute pute*: plundering and doing other things), and গুটিয়ে সুটিয়ে (*gutiye sutiye*: folding and doing similar acts) etc.

(e) **Reduplication of Verbal Nouns:** eduplication of Verbal Nouns in Bengali can be categorized as follows:

i. The base form, which is a noun, is reduplicated with a link vowel allograph -*ā*- while the second word takes the suffix marker -*i*, e.g., হাতাহাতি (*hātāhāti*: fighting with hands), লাথালাথি (*lāthālāthi*: fighting with kicks), গালাগালি (*galāgali*: embracing), and কানাকানি (*kānākāni*: whispering) etc.

ii. The first word is a verb root, which is reduplicated with a link vowel allograph -*ā*-, while the second word takes the suffix -*i*, e.g., জানা জানি (*jānājāni*: knowing), মারা মারি (*mārāmāri*: fighting), টানা টানি (*tānātāni*: pulling), ধরা ধরি (*dharādhari*: catching), and বলা বলি (*balābali*: talking) etc.

(f) **Reduplication of Adjectives:** Reduplication of Adjectives in Bengali occurs in two distinct ways:

i. The first adjectival form is reduplicated where the members have neither any suffix nor any marker attached to them, e.g., কালো কালো (*kālo kālo*: blackish), ছোট ছোট (*chota chota*: small), ভালো ভালো (*bhāla bhāla*: good), and সাদা সাদা (*sādā sādā*: white) etc.

ii. A part of the first member is duplicated while the second member remains unchanged. When the first member has no suffix or marker, the second member has the marker -*e*, e.g., ধবধবে জামা (*dhabdhabe jāmā*: pure white shirt), কুচকুচে চুল (*kuckuce cul*: jet black hair), টকটকে রং (*taktake raṅ*: dark red colour), ঘুটঘুটে অন্ধকার (*ghutghute andhakār*: pitch darkness), লকলকে জিভ (*laklake jibh*: flickering tongue),গনগনে আগুন (*gangane āgun*: burning fire) and খনখনে গলা (*khankhane galā*: shrill voice) etc.

(g) **Reduplication of Adverbs:** Reduplication of Adverbs in Bengali occurs in five distinct ways:

i. An adverbial form is duplicated where both members are free forms without any marker or suffix, such as কখনো কখনো (*kakhano kakhano*:

sometimes), হঠাৎ হঠাৎ (*hathāt hathāt*: suddenly), বার বার (*bār bār*: repeatedly), একা একা (*ekā ekā*: alone), and ক্রমে ক্রমে (*krame krame*: gradually) etc.

ii. The second member is a slightly modified form of the first member. While the first member has the ending -*ā* tagged to it, the second member has the ending -*i* attached to it, e.g., খোলাখুলি (*kholākhuli*: openly), মিছামিছি (*michāmichi*: falsely), and রাতারাতি (*rātārāti*: overnight) etc.

iii. The second member is a slightly modified form of the first member. While the first member carries the ending -*o*, the second member carries the ending -*i* with it, e.g., মুখোমুখি (*mukhomukhi*: face to face), পিঠোপিঠি (*pithopithi*: one after another in quick succession), and পুরোপুরি (*puropuri*: totally) etc.

iv. A temporal noun is reduplicated to form an adverb where both members are free forms without any marker or suffix, such as রোজ রোজ (*roj roj*: regularly), সময় সময় (*samay samay*: at certain times), and সকাল সকাল (*sakāl sakāl*: early) etc.

v. A noun word is duplicated where both forms have the marker -*āy*, e.g., বেলায় বেলায় (*belāy belāy*: before time), মাথায় মাথায় (*māthāy māthāy*: just up to the mark), and কাঁটায় কাঁটায় (*kātāy kātāy*: exactly on time) etc.

vi. An adjective is reduplicated to form an adverb where both members are free forms without any marker or suffix, such as আস্তে আস্তে (*āste āste*: slowly), প্রথম প্রথম (*pratham pratham*: initially), জোরে জোরে (*jore jore*: fast), and ধীরে ধীরে (*dhīre dhīre*: slowly) etc.

1.5 Properties of MWE

Understanding the properties of MWEs is crucial for tasks in natural language processing, linguistics, and computational language studies. Here are some key properties:

(a) **Idiomaticity:** MWEs often exhibit idiomatic usage, where the meaning of the expression cannot be derived from the individual meanings of its components. This poses a challenge, as the interpretation relies on the fixed, often non-literal, meaning of the entire expression. The meaning of an MWE may not be a straightforward composition of the meanings of its individual words. This lack of compositionality makes it challenging to process MWEs using traditional syntactic and semantic analysis methods.

(b) **Variability:** MWEs can manifest in various syntactic forms and exhibit morphological variations. Recognizing and handling these variations is challenging, as the same expression may appear in different forms depending on the context. Especially in morphologically rich languages, variability poses problems at various levels. Returning several variants of the same MWE slows down manual filtering as the same MWE is presented several times to lexicographers. For empirical methods, variability increases data sparsity.

(c) **Discontinuity:** One challenge in identification is posed by discontinuous occurrences such as look up (usage - look it up). Alien elements can intervene between core MWE components, is a challenge for MWE processing. Discovering flexible constructions, like verbal expressions, is a challenge because of their discontinuous nature.

(d) **Ambiguity:** MWEs can be ambiguous, having multiple interpretations based on the context. Disambiguating between these interpretations requires sophisticated linguistic and contextual analysis.

(e) **Frequency and Diversity:** MWEs are often highly frequent in language and cover a wide range of linguistic phenomena. Handling the sheer diversity and frequency of MWEs in large text corpora poses scalability challenges.

(f) **Recognition Across Languages:** MWEs vary across languages, and direct translation may not capture their nuances. Developing techniques that can recognize and handle MWEs across different languages is a complex task.

(g) **Dynamic Nature:** Languages evolve continuously, and new multiword expressions (MWEs) regularly emerge. Staying abreast of these changes and adapting to the dynamic nature of language use is an ongoing challenge.

1.6 Motivation of the Research

The motivation for studying reduplication stems from its crucial role in understanding the structure and function of language. Reduplication, the process of repeating all or part of a word, occurs in many languages across the world and serves various linguistic purposes. Understanding reduplication provides insights into phonological, morphological, and semantic aspects of language. Here are some key motivations for studying reduplication:

(a) **Machine Translation:** In machine translation, Multi-Word Expressions (MWEs) can pose challenges as they may have different translations compared to their individual words.

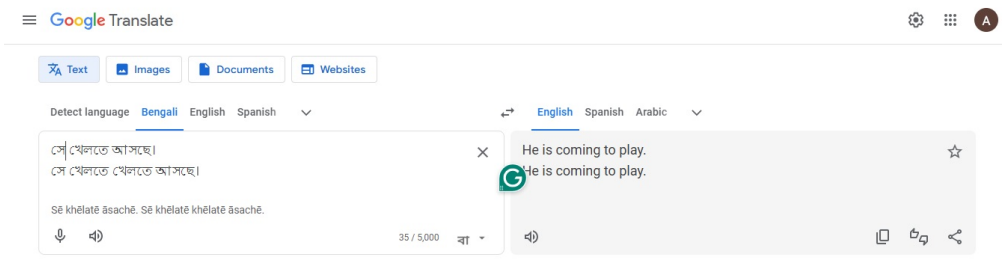

Figure 1.1: Bangali-English Google Translation (accessed on 30th December, 2024)

Figure 1.1 displays the machine translation from Bengali to English for the two sentences. While the translation of the first sentence is precise, the second sentence is inaccurately translated due to the failure to identify the reduplicated bigram **খেলতে খেলতে** (*khēltē khēltē*: while playing). This highlights the crucial role of reduplication identification in language processing applications, such as machine translation.

(b) **Word Sense Disambiguation (WSD):** Collocations aid in Word Sense Disambiguation (WSD) by providing contextual clues for disambiguating the meaning of target words. Nearby words in a collocation often offer strong and consistent hints regarding the intended sense of the target word. This contextual information enhances the accuracy of WSD systems, enabling them to assign the correct sense to ambiguous words based on their collocational context.

For example, the word **হাত** (*hāt*: hand) is polysemous and its meaning varies depending on context. However, in the partial reduplication **হাতাহাতি** (*hātāhāti*: conflict or scuffle), the combined form conveys a single, specific meaning.

(c) **Information Retrieval Systems:** Information retrieval systems aim to index phrases that are deemed ”interesting” and ”informative”. Reduplicated expressions often provide strong contextual cues, making them particularly valuable for indexing.

For example, the presence of the reduplication **আবোল তাবোল** (*abol tabol*: gibberish) in a text indicates that the following or surrounding content may not be factually reliable. Indexing such phrases can improve search accuracy and relevance.

1.7 Research Questions

To address the complexities of reduplication in Bengali and enhance the performance of Natural Language Processing (NLP) systems for various applications, this study seeks to answer the following research questions:

(a) **How can reduplicated forms in Bengali texts be accurately identified, considering the diverse linguistic structures and contextual variations?**

(b) **What methodologies can be developed to generate appropriate reduplicated forms for given base words in Bengali?**

(c) **How can semantic tags be effectively assigned to reduplicated forms, capturing their meanings within different contexts?**

By answering these questions, the research aims to develop sophisticated algorithms capable of addressing the intricacies of reduplication in Bengali, thereby contributing to the advancement of NLP for reduplication-rich languages.

1.8 Problem Definition

Natural Language Processing (NLP) systems must be capable of handling the complexities of reduplication to facilitate tasks such as machine translation, information retrieval, and text summarization. The primary objectives based on the "Research Questions" Section 1.7 are to create a system that can:

(a) Accurately identify reduplicated forms in Bengali texts.

(b) Generate appropriate reduplicated forms for given base words.

(c) Assign semantic tags to the identified reduplications, reflecting their meanings within the context.

This problem is challenging due to the diverse linguistic structures and contextual variations present in Bengali. Moreover, reduplication can appear in several forms, such as complete reduplication and partial reduplication, each requiring different handling approaches.

1.9 Research Roadmap

Each phase of the roadmap is designed to address specific aspects of "Problem Definition" Section 1.8, contributing to a comprehensive understanding and

effective handling of Bengali reduplications in NLP. The roadmap outlines the following key phases:

(a) **Data Collection:** The data collection process for building a comprehensive dataset for Bengali reduplication involves several meticulous steps to ensure the acquisition of a diverse and representative corpus. Initially, the process begins with corpus selection, targeting sources rich in natural language usage across various contexts. Literary texts, such as novels, short stories, and poems, are invaluable due to their creative and diverse use of language. News articles from newspapers and online portals provide a contemporary and formal language context.

Ethical considerations are paramount throughout the data collection process. Ensuring privacy by anonymizing or removing personal information is critical. Obtaining proper consent when collecting data from sources requiring it is also necessary to adhere to ethical guidelines.

(b) **Text Preprocessing:** Data cleaning and preprocessing are imperative before the dataset can be utilized for training and evaluation. This step includes removing noise, such as html elements and non-textual elements, from the corpus. Normalization processes standardize the text by addressing variations in spelling, punctuation, and case. Tokenization, the process of splitting the text into sentences and words, is essential for further linguistic analysis and processing. These steps ensure the dataset's quality and consistency, making it suitable for subsequent analysis.

(c) **Identification of Reduplicated Forms:** Reduplication identification refers to the process of detecting and recognizing reduplicated forms within a language. Reduplication involves the repetition of a word, part of a word, or a morpheme, and serves various linguistic purposes, such as expressing intensity, plurality, frequency, or emotional emphasis. It includes distinguishing between types, such as complete reduplication, where the entire word is repeated, and partial reduplication, involving only a segment of the word.

(d) **Generation of Reduplicated Forms:** The generation of reduplicated forms in Bengali is not just a straightforward process of duplication; it encompasses a variety of patterns and rules that reflect intricate semantic, syntactic, and phonological considerations. Understanding the underlying rules of reduplication in Bengali requires a deep dive into its morphological and syntactic structures. These variations add layers of complexity to the task of generating reduplicated forms. To generate reduplicated forms effectively,

it is essential to develop sophisticated algorithms. This involves constructing comprehensive linguistic models that incorporate rules and patterns of reduplication. This research aims to explore the methodologies for generating reduplicated forms in Bengali, examining both the theoretical frameworks that govern reduplication and the practical aspects of developing computational tools for this purpose. The ability to generate reduplicated forms can be advanced to enhance the naturalness and expressiveness of automated Bengali text, thereby improving the overall performance of NLP applications for Bengali speakers.

(e) **Sense Tagging of Reduplications:** Sense tagging of reduplications in Bengali is a crucial task in the field of natural language processing (NLP) that aims to accurately identify and annotate the meanings of reduplicated forms. Developing effective sense tagging algorithms involves the creation of comprehensive linguistic models that can handle the diverse patterns of reduplication. These models must be trained on extensive datasets to learn the various semantics associated with reduplications. Validation mechanisms are also essential to ensure that the sense tags assigned to reduplications are contextually appropriate and semantically accurate.

(f) **Validation and Testing:** Validation and testing are essential in developing robust NLP systems. Validation involves using a validation set to tune input parameters and prevent overfitting, ensuring the system generalizes well to new data. Testing evaluates the final systems's performance on a separate test set, providing an unbiased measure of its effectiveness. Evaluation Metrics from testing, such as accuracy, precision, recall, and F1-score, are crucial for assessing the model's reliability. Below are detailed descriptions and formulae for each metric, along with their applications to the tasks at hand.

- True Positives (TP): Cases where the system correctly identifies a positive instance.
- False Positives (FP): Cases where the system incorrectly identifies a positive instance.
- False Negatives (FN): Cases where the system fails to identify a positive instance.
- True Negatives (TN): Cases where the system correctly identifies a negative instance.
- Accuracy: Accuracy measures the overall correctness of the system by calculating the proportion of true results (both true positives and true

negatives) among the total number of cases examined.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: Precision measures the proportion of true positive results in relation to all positive results predicted by the system.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall: Recall, or sensitivity, measures the proportion of true positive results in relation to the total number of actual positive cases.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- F1-Score: The F1-score is the harmonic mean of precision and recall, providing a single metric that balances the two.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

1.10 Contibution and Novelty

This thesis introduces a methodology for ensuring structural uniformity in Bengali MWEs [5], crucial for efficient reduplication processing and downstream applications like text classification. It also presents an approach for eliminating contextually irrelevant words [6], thereby reducing dimensionality and improving the performance of NLP tasks such as text classification.

This study explores methods for identifying and understanding reduplications in Bengali using a blend of rule-based and machine learning approaches. A significant contribution of this study is the integration of the Levenshtein Distance—a metric for measuring the difference between two sequences—with a *Word Expansion* technique [7]. This combination enhances the precision of identifying redupli-

cated forms by comparing word similarities and expanding the linguistic scope of analysis. By crafting a hybrid system that merges these rule-based strategies with advanced machine learning algorithms, the study achieves notable improvements in the accuracy and efficiency of reduplication detection.

Beyond identification, the research explores the generative aspect of reduplications, investigating how these forms are created and structured. Finite State Transducers (FSTs) are employed as computational models to simulate the generation and identification of reduplicated forms [8]. This approach proves effective in capturing the patterns and rules governing reduplication, adding depth to the study's analytical framework.

This research explores the semantic aspects of Bengali reduplications by analyzing their contextual meanings with the help of BERT models, enabling precise interpretation. It results in the creation of a dynamic repository of Bengali reduplications and their meanings, designed to adapt and grow with new data, providing a valuable resource for NLP, machine translation, and linguistic studies.

1.11 Conclusion

This chapter lays the foundation for the research on Bengali reduplication by providing a comprehensive overview of the study's background and context. The research problem has been clearly defined, emphasizing the complexities and significance of understanding and analyzing reduplications in Bengali. Furthermore, a detailed roadmap has been outlined, guiding the systematic exploration of the research objectives and methodologies. By setting clear directions, it ensures that the research remains focused and coherent, while also paving the way for innovative solutions to the problem. Finally, the chapter highlights the unique contributions and novelty of the research. This chapter thus sets the stage for the subsequent sections of the study, where these outlined objectives and methodologies will be explored in greater detail.

CHAPTER 2

Prior Art Search

This chapter provides an extensive survey of research on the identification, generation, and sense tagging of reduplication in both international and Indian languages. It examines various approaches and methodologies used by researchers to study reduplication across different linguistic contexts, with a special focus on Indian languages where this phenomenon is particularly common. The chapter aims to offer a broad understanding of the complexities involved in studying reduplication, setting the stage for further research in this area.

2.1 Survey on Text Preprocessing

Any text corpus, compiled from numerous online sources, such as newspaper archives, books, encyclopedias, etc., contains special characters that are stop words or function words irrelevant for further processing, such as categorization, information retrieval, sense disambiguation, etc. Tokenization can be challenging when different units' lexical structures aren't always consistent. In each NLP application, preprocessing is therefore a crucial step.

(a) **Stop Word Removal:** Every natural language has a collection of stop words or function words that must be eliminated first from the text corpus before further analysis. When a predetermined list of stop words is compared to the target text that needs to be removed, it removes 13% of the words, according to Raulji et al. [9].

A stop word removal technique for the Hindi language was employed by Jha et al. [10] using the idea of a Deterministic Finite Automata (DFA) which has 99% accuracy. With the assistance of linguists, Siddiqi & Sharan [11] compiled a list of Hindi stop words containing more than 800 stop words. A rule-based approach to dynamically identifying stop words for the Gujarati language was proposed by Rakholia and Saini [12]. It produces 98.10% and

94.08% average accuracy on nearly 600 Gujarati documents, classified into routine and domain-specific categories.

Sinka and Corne [13] proposed a method to identify stop words based on word entropy. Their approach focused on the informativeness of a word, using entropy to measure the amount of information a word conveys across different documents. High entropy words, which appear with varying frequencies in different contexts, are considered informative, whereas low entropy words, which appear uniformly across documents, are likely candidates for stop words.

Al-Shalabi et al. [14] designed a Finite State Machine (FSM) specifically to eliminate stop words in the Arabic language. The FSM is a computational model used to recognize patterns within the text, allowing for the systematic removal of predefined stop words from the corpus.

Alhadidi and Alwedyan [15] developed a hybrid stop word removal technique for Arabic that combines both dictionary-based and algorithmic approaches. Their method utilizes a predefined list of stop words (dictionary-based) along with an algorithm that dynamically identifies and removes additional stop words based on their context and frequency.

Dolamic and Savoy [16] demonstrated the significant impact that stop word lists can have on the performance of information retrieval (IR) systems. By experimenting with different stop word lists, they showed how the choice of stop words can affect the efficiency and accuracy of IR tasks.

R. Puri et al. [17] created a stop word list for Punjabi by analyzing news articles from various popular Punjabi newspapers. They identified the most frequently occurring words that did not contribute meaningful information to the text, compiling these into a comprehensive stop word list for use in Punjabi text processing.

Ashish et al. [18] implemented a dictionary-based stop word elimination technique for the Gujarati language. Their approach involved comparing words in the text to a predefined list of stop words and removing any matches, thereby reducing the data size and improving processing efficiency.

Sharma and Jain [19] examined the impact of stop words on the performance of text classification tasks. They showed that removing stop words can significantly improve the accuracy and efficiency of classifiers by reducing noise in the dataset.

Additionally, Subrata and Diganta [20] created an automated technique that finds 290 stop words in Technology Development for Indian Languages (TDIL), MeitY - Govt. of India, Bengali text corpora. A machine learning-based stop word recognition system was put out by Osman et al. [21], sug-

gesting methods like the Discrete Cosine Transform (DCT) approach and feature selection with the Proportion of Variance method to exclude irrelevant phrases from the text corpus.

In addition to the general preprocessing activities, Abhijit and Diganta [6] introduced Inverse Class Frequency (ICF)-based filtering of domain-specific unimportant terms. Applying this preprocessing technique, they achieved 95.92% accuracy on text classification on Technology Development for Indian Languages (TDIL), MeitY - Govt. of India, Bengali text corpora. In this paper, a specialized preprocessing technique on duplicated MWE is proposed to improve text classification on the same dataset.

These studies illustrate the diverse methodologies employed across different languages to identify and remove stop words, each contributing to more efficient and accurate text processing and analysis.

(b) **Lemmatization and Stemming:** Lemmatization and stemming are two crucial preprocessing techniques in natural language processing (NLP). Both techniques aim to reduce words to their base or root forms, thereby simplifying the text for further analysis.

Lemmatization has been shown to improve the performance of NLP tasks significantly. For instance, Chakrabarty et al. [22] demonstrated enhancements in Information Extraction (IE) and Information Retrieval (IR) systems attributed to the increased lexical coverage facilitated by lemmatization. They proposed a neural network-based lemmatization method and demonstrated its effectiveness for Bengali, a highly inflected language. BenLem [23], a lemmatizer for Bengali, plays a crucial role in languages with complex morphology, especially for tasks like sense disambiguation. Lindén and Pirinen [24] illustrated the effectiveness of lemmatization in Finnish, while Darwish [25] highlighted its importance in Arabic. Sharoff [26] emphasized the need for lemmatization in Russian to achieve high-quality text analysis.

Stemming is particularly useful for search engines and information retrieval tasks, where speed is critical. Jivani [27] provided a comparative study of stemming algorithms, demonstrating their effectiveness in various NLP applications. Stemming techniques have been adapted for different languages. For example, Kraaij and Pohlmann [28] developed stemming algorithms for Dutch, while Savoy [29] adapted the Porter Stemmer for French. These adaptations highlight the importance of language-specific rules in stemming. In Bengali, there is a rule-based stemmer [30] that utilizes the Bengali WordNet [31].

Recent advancements in NLP, such as the development of transformer

models like BERT [32] and GPT-3 [33], have incorporated lemmatization and stemming into their preprocessing pipelines. These models demonstrate the continued relevance and importance of these techniques in modern NLP.

(c) **Parts-of-Speech (POS) Tagging:** It is a fundamental preprocessing step in natural language processing (NLP) that involves labeling each word in a text with its corresponding part of speech. POS tagging is essential for various downstream NLP tasks. It helps in syntactic parsing, semantic analysis, and improving the accuracy of machine translation systems.

Early POS tagging methods relied on handcrafted rules. These methods use a set of linguistic rules to assign POS tags based on the word's context and morphological features. The Brill tagger [34] is a well-known example of a rule-based POS tagger.

Statistical methods employ probabilistic models trained on annotated corpora. Hidden Markov Models (HMMs) and Maximum Entropy models are popular statistical approaches. For instance, Ratnaparkhi [35] developed a Maximum Entropy-based POS tagger that showed significant improvements over rule-based methods.

With advancements in machine learning, algorithms such as Support Vector Machines (SVMs) and Conditional Random Fields (CRFs) have been applied to POS tagging. The use of CRFs, as shown by Lafferty et al. [36], allows for the incorporation of various contextual features, improving tagging accuracy.

Recently, deep learning models, including Recurrent Neural Networks (RNNs) and Transformers, have achieved state-of-the-art performance in POS tagging. For example, Huang et al. [37] demonstrated that BiLSTM-CRF models outperform traditional machine learning methods. Additionally, the BERT model [32] has been fine-tuned for POS tagging, achieving remarkable results. Recent advancements include the integration of contextualized word embeddings and transfer learning. Models like ELMo [38] and BERT [32] provide deep contextual representations, improving POS tagging accuracy.

English POS tagging is well-established, with numerous annotated corpora and tools available. The Penn Treebank [39] is a widely used resource. Chinese POS tagging involves handling word segmentation simultaneously. The work by Xue and Shen [40] addresses these challenges using a combined segmentation and tagging approach. Arabic's rich morphology and diacritics pose challenges. Habash and Rambow [41] developed a morphological analyzer and tagger specifically for Arabic. Indian languages like Hindi and Bengali have complex morphology. Dandapat et al. [42] created POS taggers

for Bengali, achieving high accuracy using CRFs.

2.2 Survey on MWE Identification

2.2.1 Statistical Approaches

Statistical methods are employed to assess the compositionality of Multi Word Expressions (MWEs). Compositionality, in this context, refers to the idea that "the meaning of the whole can be strictly predicted from the meaning of the parts" [43]. In contrast, decomposability measures the extent to which the meaning of an MWE can be attributed to its individual parts [44]. For instance, if the bi-gram $<w^1w^2>$ is considered an MWE, decomposability evaluates the degree to which the meanings of w^1 and w^2 contribute to the overall meaning of $<w^1w^2>$. The compositionality factor is essentially the reciprocal of decomposability. Here are some key statistical metrics commonly used for this purpose:

(a) **Frequency Based Method [43]**: The approach described is a straightforward and easily implementable method for extracting collocations, particularly suited for lightweight computations. The fundamental idea is to identify the most frequent bi-grams in a given corpus. However, this basic approach has a limitation: it tends to retrieve many insignificant bi-grams, such as common phrases like "of-the" or "in-the". To enhance the method's effectiveness, a simple heuristic is introduced. The candidate phrases, identified as frequent bi-grams, undergo additional processing using a Parts-of-Speech (POS) tagger. Only combinations with a high probability of being meaningful phrases are considered for further analysis. This refinement helps filter out irrelevant and common combinations, improving the quality of the extracted collocations. The POS tag structures taken into account in this process include AN (Adjective-Noun), NN (Noun-Noun), AAN (Adjective-Adjective-Noun), ANN (Adjective-Noun-Noun), NAN (Noun-Adjective-Noun), NNN (Noun-Noun-Noun), and NPN (Noun-Preposition-Noun). This selection allows the method to focus on specific syntactic patterns that are likely to represent meaningful collocations. Overall, this approach balances simplicity with effectiveness in identifying relevant and contextually meaningful bi-grams.

(b) **Mean and Variance [43]**: The previously mentioned frequency-based method is suitable for identifying fixed phrases, but it may not effectively capture the relationships between words that exhibit a flexible or variable-length structure. Consider instances where certain words frequently co-occur but can have varying numbers of words between them. For instance, the colloca-

tion "knock...door" remains relevant even if there are several words between "knock" and "door" in a sentence. In such cases, "knock" consistently serves as the verb associated with "door." To address this variability, the proposed method involves calculating both the mean and variance of the distances between two words. This statistical approach allows for a more flexible identification of collocations, accommodating instances where words maintain a significant relationship despite variations in the number of intervening words. By considering the mean and variance of distances, the method captures the inherent variability in word relationships, enhancing its ability to detect meaningful collocations with variable lengths.

(c) **Hypothesis Testing [43]**: In the context of Hypothesis Testing , the null hypothesis, denoted as H_0, posits that the words w^1 and w^2 are independent of each other. This is expressed as:

$$H_0 : P(w^1, w^2) = P(w^1) \cdot P(w^2)$$

Here, $P(w^1, w^2)$ represents the joint probability of w^1 and w^2, while $P(w^1)$ and $P(w^2)$ denote their respective marginal probabilities.

Various statistical tests, such as the t-test and Pearson's chi-square test, can then be employed to either accept or reject the null hypothesis. The t-test is particularly useful when the null hypothesis assumes that the sample is drawn from a normal distribution, as is often the case in Bernoulli trials. On the other hand, when the distribution is not normal, Pearson's chi-square test becomes a valuable alternative.

(d) **Likelihood Ratio [43]**: The Likelihood Ratio [43] offers an alternative approach to hypothesis testing, providing a numerical indication of how much more likely one hypothesis is compared to another. In this method, two hypotheses are considered:

1. Hypothesis 1: w^1 and w^2 are independent, expressed as $P(w^1|w^2) = p = P(w^1|\neg w^2)$.

2. Hypothesis 2: w^1 and w^2 are not independent, indicated by $P(w^1|w^2) = p^1 \not\approx p^2 = P(w^1|\neg w^2)$.

(e) **Pointwise Mutual Information (PMI) [43]**: PMI is a measure derived from information theory to calculate the mutual information between two events. In the context of computational linguistics, PMI is applied to assess the compositionality of pairs of words, w^1 and w^2.

The formula for PMI between w^1 and w^2 is expressed as:

$$PMI(w^1, w^2) = \log\left(\frac{P(w^1, w^2)}{P(w^1) \cdot P(w^2)}\right)$$

Here, $P(w^1, w^2)$ represents the joint probability of w^1 and w^2, while $P(w^1)$ and $P(w^2)$ denote their individual probabilities.

(f) **Other Methods**: The Fuzzy Bigram Index (FBI) [45] employs Fuzzy Set theory to assign a compositionality value within the range of [0,1] to a bigram within the Bengali text corpus. The Statistical measure Salience [46] is applied specifically to Bengali Noun-Verb (N-V) Collocations, treating them as Multi-Word Expressions. Salience relies on a combination of PMI and the Frequency of w^1 and w^2. Another approach to assess the compositionality of Multi-Word Expressions involves using nearest neighbors in vector space models [47]. This method calculates the average distance between a phrase vector and its substituted phrase vectors, with the constitutionality being proportional to this distance. Additionally, a generic approach [48] for extracting MWEs is developed based on Dependency Trees. It constructs a dependency tree for a sentence and identifies whether a sub-tree is an MWE or not, relying on Entropy. Other measures such as Dice's coefficient, Z score, and Selectional association are also employed to evaluate the compositionality of MWEs.

2.2.2 Rule Based Approaches

Rule-based approaches leverage linguistic rules to identify MWEs based on lexical patterns, syntactic structures, or morphological features. According to Niladri Sekhar Dash [49], the formation of compound words stands out as one of the most prolific methods of word creation in Bengali, significantly contributing to the language's lexical expansion. This process plays a diverse role in enriching the vocabulary of Bengali.

In the context of Bengali, the identification of verb-verb compound words involves the application of several syntactic[50] and semantic rules [51]. These rules are specifically designed to detect the presence of compound words formed by combining verbs. Additionally, there are rules dedicated to detecting named entities or proper nouns [52] within the linguistic context. Moreover, regular expressions [53] have been formulated and employed for the purpose of identifying Multi-Word Expressions (MWEs). These expressions are essentially patterns defined in a rule-based manner to recognize specific combinations of words that form meaningful and unitary expressions in Bengali. The utilization of these rules and regular expressions contributes to the systematic identification and extraction of compound

words and MWEs in the Bengali language.

Dutta and Jindal [54] introduced a word similarity-based algorithm to detect reduplication within a Hindi corpus. Their approach involved translating Hindi MWE word pairs into English, subsequently measuring the degree of similarity between the translated pairs. Nongmeikapam and Bandyopadhyay [55] proposed a rule-based mechanism for identifying partial reduplication in Manipuri. The method entails character-wise comparisons, both from the front and rear, for words, as well as assessing the partial reduplication through character matches between the last characters of the first word and the initial characters of the second word. Experimental results demonstrated the approach's efficacy, yielding an overall average Recall, Precision, and F-Score of 94.24%, 82.27%, and 87.68% respectively.

P. Garg et al. [56] introduced a word similarity-based technique for detecting reduplication in Punjabi. Their algorithm incorporated word length as an evaluation criterion, achieving an accuracy rate of 85%.

2.2.3 Machine Learning Based Approaches

In the domain of Natural Language Processing (NLP), the identification of Named Entities and Noun-Noun (N-N) bigrams holds significant importance for various language processing tasks, including information extraction, sentiment analysis, and machine translation. One prominent method involves the utilization of machine learning algorithms such as Random Forest, as demonstrated by Gayen and Sarkar [57]. They proposed a method for determining Named Entities or Noun-Noun bigrams from Bengali text corpora. In their approach, they first parse the raw Bengali text using a Bengali Shallow Parser [1] to extract N-N bigrams. Subsequently, statistical features such as pointwise mutual information (pmi), salience, log likelihood, chi-square, and t-score, along with syntactic and linguistic features, are employed to train a Random Forest model. This model is then used to automatically identify Named Entities or Noun-Noun bigrams in the text corpus. Their method achieved an impressive F-Score of 85.20%, indicating its effectiveness in identifying these linguistic structures.

In addition to machine learning approaches, semantic clustering-based methods have also been proposed for identifying bigram noun-noun Multi-Word Expressions (MWEs) from text corpora. Tanmoy Chakraborty [58] presented a novel approach based on semantic clustering to identify bigram MWEs in a medium-sized Bengali corpus. To measure this similarity, they utilize synonymous sets of the component words. By leveraging semantic clustering techniques, they provide

[1] http://ltrc.iiit.ac.in/analyzer/bengali/

a unique perspective on MWE identification in Bengali text corpora.

Furthermore, Pavel Pecina [59] explored the use of various machine learning techniques, including linear logistic regression, linear discriminant analysis (LDA), and Neural Networks, for extracting Multi-Word Expressions (MWEs) from text corpora. Pecina's approach involves the construction of a feature vector comprising 55 association measures, capturing the contextual information surrounding each MWE. By employing advanced machine learning techniques and feature engineering, Pecina demonstrates the effectiveness of his method in accurately identifying MWEs.

2.2.4 Deep Learning Based Approaches

Gharbieh et al. [60] made significant strides in the field of Multi-Word Expression (MWE) identification by developing the first deep learning models, including Long Short-Term Memory (LSTM) Networks, Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN), designed for broad-coverage MWE identification tasks. Their pioneering work laid the foundation for subsequent research in this domain, demonstrating the efficacy of deep learning approaches in automatically identifying MWEs in textual data. Their dataset, meticulously annotated with eight pieces of information for each token, provided a rich resource for training and evaluating MWE identification models. The inclusion of details such as the numeric position of the token in its sentence, the token itself, its lemmatized form, part-of-speech tag, gold-standard MWE tag, the position of the last token in its MWE, supersense tag, and sentence ID facilitated the development of robust and accurate models. In their experiments, Gharbieh et al. [60] utilized TensorFlow version 0.12 to implement the neural network models and scikit-learn for traditional machine learning models. The experiments were conducted on 2 GHz Intel Xeon E7-4809 v3 CPUs, showcasing the scalability and efficiency of their proposed approaches. The CNN3 model achieved the highest F-score on the test set, reaching an impressive 59.96%, indicating the effectiveness of deep learning techniques in MWE identification tasks.

To address the challenge of detecting discontinuous MWEs, such as "put one of the blue masks on", a Graph Convolutional Network (GCN) preceding Bidirectional Long Short-Term Memory (Bi-LSTM) Networks was employed. This architecture proved effective in capturing complex MWE structures and achieved notable performance improvements in identifying discontinuous expressions.

Furthermore, leveraging contextualized token-based embeddings such as ELMo embeddings [38] enhanced the representation of words in their respective contexts. Unlike traditional type-based word representations like word2vec or GloVe,

token-based embeddings better capture the syntax and semantics of each word, improving the overall performance of MWE identification models.

In addition to the advancements in deep learning techniques, transformer-based models like BERT have revolutionized NLP tasks by generating context-sensitive embeddings. BERT's versatility has been demonstrated across various NLP tasks, including question answering [61], named entity recognition [62], and sentiment analysis [63]. Researchers have successfully applied multilingual BERT embeddings to analyze sentiments in languages such as Bengali [64], [65], highlighting the model's adaptability to diverse linguistic contexts.

The continuous evolution of transformer-based models has led to the development of several variants and extensions, each aimed at improving performance and addressing specific challenges. For example, RoBERTa (Robustly optimized BERT approach) [66] and ALBERT (A Lite BERT) [67] have introduced enhancements such as training with more data, parameter-sharing techniques, and self-supervised learning to further optimize model performance. These advancements underscore the ongoing innovation in transformer-based architectures and their growing impact on NLP research.

2.3 Reduplication Generation

Reduplication, a widespread linguistic phenomenon present in approximately 85% of the world's languages, remains a fascinating yet challenging area of study. Despite its ubiquity, the automatic identification and modeling of reduplication patterns have not received comprehensive documentation in the literature [68]. This gap in research highlights the need for further exploration and development of computational methods for analyzing reduplication.

One promising approach to modeling reduplication is the use of Finite State Transducers (FSTs), which have shown efficacy in capturing the intricate patterns of reduplicative morphology across languages [69][70][71]. Beesley and Karttunen [69] were among the pioneers in proposing the use of FSTs for linguistic morphology, laying the groundwork for subsequent research in this area. Dolatian and Heinz [70] further expanded on this work by demonstrating the versatility of 2-way FSTs in handling various reduplicative patterns.

In addition to FSTs, recent advancements in computational linguistics have explored the application of neural network models for modeling reduplication. For example, Rajpurkar et al. [61] proposed a deep learning approach to reduplication modeling, leveraging the power of recurrent neural networks (RNNs) to capture sequential patterns in reduplicated forms. Similarly, Lample et al. [62] introduced a novel neural architecture for modeling morphological processes, including

reduplication, in a data-driven manner.

The distinction between 1-way and 2-way FSTs is crucial in the context of modeling reduplication. While 1-way FSTs are limited to unidirectional movement along the input tape, 2-way FSTs offer bidirectional movement, allowing for a more comprehensive analysis of reduplicative structures [68]. This bidirectional capability enables the FST to capture both the prefix and the base forms of reduplicated words, facilitating a more accurate understanding of reduplication patterns. Recent studies [72], [73] have utilized pre-trained language models (LMs) for identifying multiword expressions (MWEs) in various languages, including French and Irish.

It is observed that various rule-based, machine learning, and deep learning approaches have been developed for the identification of republications in several natural languages. A few recent research efforts have focused on modeling the reduplication formation process using finite-state machines. However, there have been no attempts to design a computational model specifically for both partial and complete reduplications in Bengali or to use such models for their identification. This research focuses on modeling partial and complete reduplication in Bengali using two-way finite-state transducers, which are configured to represent the language's specific patterns. These practical machine models are employed to identify reduplication instances in Bengali texts, enhancing the study's applicability and providing valuable insights into Bengali language analysis.

2.4 Sense Tagging of Reduplication

Sense tagging, or word sense disambiguation (WSD), is the process of assigning the correct sense or meaning to a word based on its context in a text. This task is particularly challenging in languages with rich morphological structures like Bengali.

Recent studies have focused on developing methods for sense tagging in Bengali. Researchers [74][75] have employed various methodologies, including rule-based approaches as well as supervised and unsupervised machine learning techniques, to achieve this goal. For instance, Abdulgabbar et al. [74] utilized Wikipedia to build a sense-tagged corpus, which was then used for supervised WSD. Similarly, Pal et al. [75] proposed a novel supervised methodology for WSD in Bengali, demonstrating significant improvements over previous approaches.

While these methods have shown promising results, sense tagging at the expression level, which involves capturing the meanings of Multi-Word Expressions (MWEs), remains a relatively unexplored area of research. MWEs often carry idiomatic or context-specific meanings that are not directly inferable from the

individual words they comprise. Addressing this gap could lead to significant advancements in the processing and understanding of Bengali text, providing more accurate and contextually relevant language models.

Recent advancements in neural network architectures and the availability of large annotated corpora have opened new avenues for research in WSD. For instance, Dolatian and Heinz [70] modeled reduplication with finite-state transducers, demonstrating a novel approach to handling complex linguistic phenomena. Similarly, Filiot and Reynier [71] explored the use of transducers, logic, and algebra for functions of finite words, which could be applied to improve WSD systems.

Moreover, the integration of BERT with existing WSD methodologies offers a promising direction for future research. By leveraging BERT's contextual embeddings, it is possible to develop more sophisticated WSD systems that can accurately disambiguate words based on their specific context.

BERT (Bidirectional Encoder Representations from Transformers) [76][77] represents a groundbreaking advancement in unsupervised language representation models. Unlike traditional models such as Word2Vec or GloVe [78], which generate a single static embedding for each word, BERT leverages the transformer architecture to create dynamic word embeddings that consider the context of each word occurrence within a sentence.

The transformer architecture, introduced by Vaswani et al. [79], is fundamental to BERT's performance. This architecture utilizes an attention mechanism that allows the model to weigh the importance of different words in a sentence, thereby capturing the complex dependencies and relationships between them. This mechanism enables BERT to understand the meanings of words based on their surrounding context.

BERT's pre-training involves two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, random words in a sentence are masked, and the model learns to predict these masked words based on the context provided by the other words in the sentence. NSP, on the other hand, trains the model to understand the relationship between two sentences. After pre-training on a large corpus, BERT can be fine-tuned on specific downstream tasks with relatively small amounts of task-specific data, demonstrating remarkable adaptability and performance across various NLP applications [32].

The contextual embeddings produced by BERT are a significant departure from the static embeddings of previous models. While models like Word2Vec [78] and GloVe [80] provide a single embedding for each word regardless of its context, BERT generates embeddings that vary depending on the word's context in a sentence. This allows BERT to capture the polysemous nature of words, where a single word can have multiple meanings depending on its usage.

BERT's ability to generate context-sensitive embeddings has been leveraged in numerous NLP tasks such as question answering, named entity recognition, and sentiment analysis. For example, researchers have utilized multilingual BERT embeddings to analyze sentiments in the Bengali language [64][65]. This application demonstrates BERT's versatility and effectiveness in handling languages beyond English, highlighting its potential for broader linguistic research.

Moreover, the integration of BERT with existing WSD methodologies offers a promising direction for future research. By leveraging BERT's contextual embeddings, it is possible to develop more sophisticated WSD systems that can accurately disambiguate words based on their specific context. This approach has already been explored in languages like English [81] and Spanish [82], and similar techniques can be adapted for Bengali.

Given the diversity of languages and the necessity for cross-lingual NLP applications, multilingual BERT models have been utilized to perform WSD across different languages. For instance, Wu and Dredze [83] demonstrated the effectiveness of multilingual BERT in cross-lingual tasks without needing explicit cross-lingual supervision. This capability is particularly beneficial for low-resource languages like Bengali, where annotated corpora are limited.

Integrating WSD with other NLP tasks, such as machine translation [84], information retrieval [85], and text summarization [86], can enhance the overall performance of these applications. For example, accurate sense tagging can significantly improve the quality of machine translation systems by ensuring that the correct meanings of words are translated in context. Similarly, in information retrieval, understanding the precise meaning of query terms can lead to more relevant search results.

2.5 Conclusion

This chapter provided a comprehensive review of existing research in the domain of text preprocessing and Multi-Word Expression (MWE) identification, with a specific focus on Bengali reduplication. The survey covered various approaches to MWE identification, including statistical methods, rule-based techniques, machine learning approaches, and deep learning models. Additionally, key complexities associated with reduplication generation and sense tagging were discussed, emphasizing the foundation for capturing linguistic variations in computational models. This review helps in building a strong foundation for the research, aiming to develop better tools for processing Bengali text accurately.

CHAPTER 3

Text Preprocessing

Preprocessing is a crucial step in Natural Language Processing (NLP) that involves preparing and cleaning text data for analysis and model training. This chapter focuses on preprocessing techniques for Bengali text, addressing unique challenges posed by its script, grammar, and lexical characteristics. Bengali, an Indo-Aryan language spoken primarily in Bangladesh and India, uses a syllabic alphabet that requires specialized tokenization to handle its complex characters and consonant-vowel combinations. The language's grammar, with its compound verbs, complex sentence structures, and inflectional morphology, necessitates adapted stemming and lemmatization techniques. Additionally, the rich set of synonyms, homonyms, and regional dialect variations in Bengali calls for meticulous text normalization and ambiguity resolution. This chapter provides a comprehensive guide to effectively preparing Bengali text for NLP tasks by addressing these specific challenges.

Contribution: This thesis presents a methodology to maintain structural uniformity in Bengali MWEs [7], essential for effective reduplication processing and downstream tasks like text classification. Additionally, it proposes an approach to remove contextually irrelevant words [6], which reduces dimensionality and enhances the performance of NLP tasks such as text classification.

3.1 Data Collection

The data collected from various sources for the study of Bengali reduplication identification is thoroughly described below:

(a) **Bengali Named Entity and Multi Word Expression List-CLIA**: This dataset has been meticulously compiled by the Ministry of Electronics and Information Technology (MeitY) under the Government of India[1].

[1] `https://www.meity.gov.in/`

i. It contains 899 instances of Multi-Word Expressions (MWEs), with 858 appearing as bigrams featuring a gap between the two constituent words.

ii. Additionally, the dataset includes 126 cases of complete reduplication and 21 instances of partial reduplication, showcasing a diverse range of linguistic structures. It is referred to as the *Reduplication List* in this study.

(b) **Bengali Corpus from TDIL**: This corpus is sourced from the Technology Development for Indian Languages Programme (TDIL) under MeitY - Government of India.

i. It encompasses over ten thousand sentences containing reduplications, spanning various domains such as Accountancy, Botany, Geography, Literature, Religion, Veterinary, and Zoology.

ii. The corpus consists of 1,374 documents, containing a total of 280,000 sentences and 3,347,000 words, making it a comprehensive resource for analyzing reduplication patterns.

(c) **POS Tagged Bengali Corpus from TDIL**: This corpus is part of the Technology Development for Indian Languages Programme (TDIL) under MeitY - Government of India.

i. The corpus is sourced from various Bengali newspapers and includes news articles on Sports, Tourism, Politics and Public Administration, Literature, Arts and Culture, Entertainment, Economy, and Agriculture.

ii. It contains a total of 49 documents and features 127,605 unique words or terms.

(d) **Bengali Corpus from Wikipedia**: Data collected from Bengali Wikipedia[2] supplements the research with a rich and diverse dataset.

i. This collection includes 432,000 sentences and 6,027,000 words, providing a substantial addition to the corpus.

ii. Among these, over five thousand sentences feature instances of reduplications, further enriching the dataset with examples from various contexts and domains.

These datasets collectively provide a robust foundation for the study of Bengali reduplication identification, offering a wide array of linguistic structures and contexts to analyze and understand reduplication patterns in the Bengali language.

[2] `https://bn.wikipedia.org/wiki/`

3.2 Generic Preprocessing

Generic preprocessing tasks are an integral part of the Natural Language Processing (NLP) pipeline and are usually performed before both reduplication tagging and downstream processing. These tasks include operations such as text normalization, tokenization, and the removal of noise, such as special characters, extraneous spaces, and non-linguistic elements. The primary objective of these preprocessing steps is to prepare the raw text for efficient analysis by ensuring its consistency and structure. By addressing inconsistencies and standardizing the input, these tasks lay the foundation for accurate reduplication tagging, which relies on a clean and well-processed text to identify and annotate reduplicated forms effectively. Furthermore, performing these preprocessing tasks upfront enhances the quality and reliability of subsequent NLP tasks, such as text classification, sentiment analysis, and information retrieval, by reducing unnecessary computational overhead and ensuring that only meaningful data is passed through the pipeline.

3.2.1 Removal of Special Characters

The corpus used in this study includes a variety of special characters, such as *<html>* *_* *”* *?* *(* *)* *‘* *’* *:* *.* *!* *;* etc. These characters frequently appear in Bengali literature due to the inclusion of modern formatting, punctuation styles, and digital transcription practices. However, from the perspective of Natural Language Processing (NLP), such characters are often considered *noise* that can hinder the performance of text analysis tasks.

For instance, the presence of `<html>` tags typically indicates metadata or structural components of a digital document rather than meaningful linguistic content. Similarly, punctuation marks like `!`, `?`, and `;` may contribute to syntactic structure but do not always convey significant semantic information required for specific computational tasks.

To enhance the accuracy and efficiency of text processing, it is essential to preprocess the corpus by removing these special characters. This cleaning step ensures that the text is free from irrelevant elements, thereby reducing computational complexity and improving the quality of downstream NLP tasks.

3.2.2 Tokenization

Tokenization is the process of dividing text into smaller units known as tokens, which are typically words or sub-words in natural language processing (NLP). This step is crucial for various NLP tasks such as text processing, language modeling,

and machine translation. Tokenization involves splitting a string or text into a list of tokens, where each token can be a word or sub-words in a sentence or a sentence in a paragraph.

The process uses a tokenizer to segment unstructured data and natural language text into distinct chunks, treating them as separate elements. These tokens are converted into vectors, transforming unstructured text documents into numerical data structures suitable for machine learning. This conversion allows immediate utilization of tokenized elements by a computer for practical actions and responses. Additionally, tokens can serve as features within a machine learning pipeline, enhancing decision-making processes and behaviors.

Tokenization is categorized into various types depending on how the text is segmented. Here are several classifications of tokenization:

(a) **Sentence Tokenization:** Sentence tokenization, also known as sentence segmentation, is the process of dividing a text into its individual sentences. In natural language processing (NLP), this technique is used to split a paragraph or document into its constituent sentences. The goal of sentence tokenization is to identify the boundaries between sentences. In Bengali, sentence boundaries are commonly marked by specific punctuation symbols such as `।`, `?`, `!`, and `॥`. Among these, `॥` is often used in poetic or formal contexts to delineate ideas or separate verses. This process enables NLP models to analyze and process text at the sentence level, which is often necessary for tasks such as sentiment analysis, machine translation, and text summarization.

(b) **Word Tokenization:** Word tokenization, also referred to as word segmentation, is the process of dividing a text into its individual words or tokens. In natural language processing (NLP), word tokenization is a fundamental preprocessing step that breaks down a sentence or document into its constituent words, punctuation marks, or other meaningful units. The goal of word tokenization is to create a standardized representation of text that can be easily processed by NLP algorithms and models.

(c) **Subword Tokenization:** Subword tokenization is a technique that breaks down words into smaller sub-units or sub-words. This method is particularly useful for handling complex words, compound words, and morphologically rich languages like Bengali. As an illustration, Bengali terms like পড়াতে (*paṛate*: to read) are deconstructed into two sub-words: পড়া (*paṛa*) and তে (*te*).

(d) **Character Tokenization:** Character tokenization is a process of dividing a text into individual characters. Each character, including letters, digits, punctuation marks, and whitespace, is treated as a separate token. This

approach breaks down the text at the character level, creating tokens that represent each individual unit in the text. Character tokenization is useful in scenarios where word boundaries are not clearly defined or when working with languages that do not use whitespace as a delimiter between words.

(e) **N-gram Tokenization:** N-gram tokenization is a technique used to break down text into contiguous sequences of n items, where an item can be a character, word, or subword. These sequences, known as n-grams, capture the local structure and context of the text by representing consecutive groups of items. For instance, in word-level n-gram tokenization, a Bengali sentence such as সে খেলতে খেলতে আসছে (*se khelte khelte āsche*: He is coming while playing) can be tokenized into bi-grams (2-grams) as {সে খেলতে, খেলতে খেলতে, খেলতে আসছে}, and tri-grams (3-grams) as {সে খেলতে খেলতে, খেলতে খেলতে আসছে}, and so forth. This approach captures the sequential relationships between words and is particularly beneficial for tasks such as language modeling, syntactic analysis, and contextual understanding. N-gram tokenization is commonly used in natural language processing tasks such as language modeling, and Multi-Word Expression processing, where it helps capture the sequential dependencies and patterns in text data.

3.3 Specialized Preprocessing

Specialized preprocessing techniques are tailored to address specific requirements and are not universally suitable for all tasks in Natural Language Processing (NLP). Unlike generic preprocessing steps, which are broadly applicable and focus on tasks such as tokenization and noise removal, specialized preprocessing is designed to handle task-specific challenges. Similarly, domain-specific preprocessing, such as the handling of medical jargon or legal terminologies, is necessary in specialized applications but irrelevant in general text classification tasks. Therefore, the application of specialized preprocessing must be carefully aligned with the objectives and constraints of the target task to ensure optimal performance and accuracy.

3.3.1 Maintain Structural Uniformity of MWEs

One fundamental aspect of NLP preprocessing is the standardization of spelling. Ensuring consistent spelling is critical because it enhances the accuracy of subsequent text analysis processes. A specific challenge within this context is the handling of Multi-Word Expressions (MWEs). These expressions often appear in text documents with various structural forms. For instance, there might be

spaces or hyphens ('-') between the constituent words of an MWE, or in some cases, the delimiters between words might be entirely absent. This variability can complicate the processing of MWEs, as different forms of the same expression may be treated as distinct entities by NLP algorithms. Due to orthographic inconsistencies, reduplicated words are often written with or without spaces between the repeated parts. For example, সামনা সামনি (*sāmanā sāmani*: face to face) appears without spaces on the list of MWEs published by TDIL. However, variations such as সামনা-সামনি (*sāmanā-sāmani*) with hyphens ('-') as delimiters, and সামনাসামনি (*sāmanāsāmani*) without any delimiters, are both found on the web.

To address this challenge, it is necessary to normalize the structural forms of MWEs. Normalization involves converting the different structural variants of an MWE into a single, standardized form. This step is crucial for ensuring that the NLP system can accurately recognize and process MWEs, regardless of how they are presented in the text.

3.3.1.1 Definitions and Notations

The definitions and notations below are utilized in the methodology pertain to maintain structural uniformity of MWEs in a Bengali text corpus.

Substring: Let a string s be given as $s = c_1c_2 \dots c_m \dots c_kc_{k+1} \dots c_n$. The operation to extract a substring or subsequence from the string s is defined as follows:

$$\text{substring}(s, m, k) = c_m \dots c_k \quad \text{where } 1 \leq m \leq k \leq n \tag{3.1}$$

In this operation, the substring consisting of characters from position m to position k within the string s is selected.

String Length: The length of a string is defined as the number of characters present in the string. Let a string s be given as $s = c_1c_2 \dots c_n$. The operation to determine the length of the string s is defined as follows:

$$\text{length}(s) = n \tag{3.2}$$

Here, the length of the string s is calculated by counting the total number of characters from the first character c_1 to the last character c_n.

Levenshtein distance: The Levenshtein distance, also known as minimum edit distance (MED), is a metric used to measure the similarity between two strings by calculating the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into the other.
The Levenshtein distance, also known as edit distance, between two strings s and

t (of length $|s|$ and $|t|$ respectively) can be calculated using the following formula:

$$\text{lev}(s,t) = \begin{cases} |s| & \text{if } |t| = 0 \\ |t| & \text{if } |s| = 0 \\ \text{lev}(\text{tail}(s), \text{tail}(t)) & \text{if } s_1 = t_1 \\ \min \left\{ \begin{array}{l} \text{lev}(\text{tail}(s), t) + 1 \\ \text{lev}(s, \text{tail}(t)) + 1 \\ \text{lev}(\text{tail}(s), \text{tail}(t)) \end{array} \right\} & \text{otherwise} \end{cases} \tag{3.3}$$

where tail(s) and tail(t) denote the substrings obtained by removing the first character of s and t, respectively.The notation s_1 and t_1 represent the first characters of strings s and t respectively.
For example, in the word খাওয়া ($khāwā$), If খ(kha) is replaced by দ(da) then খাওয়া is transformed to দাওয়া ($dāwā$). So, the Levenshtein distance between খাওয়া ($khāwā$) and দাওয়া ($dāwā$) is 1, i.e lev(খাওয়া, দাওয়া)=1.

3.3.1.2 Algorithm for Structural Uniformity

In this algorithm, *unify_mwe*, a word is initially divided into two subsequences of characters. The minimum edit distance between these two subsequences is then computed. If the minimum edit distance is 0, it indicates total reduplication. If the minimum edit distance is 1, it indicates partial reduplication. When the minimum edit distance is greater than 1, the word is classified as either a non-reduplicated Multi-Word Expression (MWE) or a whole word. If both subsequences are found in the dictionary[3], the result is identified as an MWE; otherwise, it is considered a whole word.

The details of all internal variables for each iteration of the *unify_mwe* algorithm are documented in Table 3.1. The step-by-step execution of the algorithm for the input strings একচোখা ($ēkacōkhā$) and খাওয়াদাওয়া ($khāwādāwā$) is described in Table 3.1.

Let, w = একচোখা be the input to the algorithm *unify_mwe*. In the first iteration, w = একচোখা is divided into two parts: w^1 = এ and w^2 = কচোখা . The minimum edit distance between w^1 = এ and w^2 = কচোখা is 5, and neither of these parts is found in the dictionary. In the second iteration, w^1 = এক and w^2 = চোখা . The minimum edit distance between w^1 = এক and w^2 = চোখা is 4, and both words are present in the dictionary. Thus, এক চোখা is returned as a Multi-Word Expression (MWE).

[3]https://en.wiktionary.org/

Algorithm 1: unify_mwe

Input: w (a sequence of characters without spaces)
Output: $< w^1\ w^2 >$
Initialization:
$n \leftarrow \text{length}(w)$
for $index \leftarrow 1$ ***to*** $n-1$ **do**
  $w^1 \leftarrow \text{substring}(w, 1, \text{index})$
  $w^2 \leftarrow \text{substring}(w, \text{index}+1, n)$
  if $lev(w^1, w^2) = 0$ **then**
    return $< w^1\ w^2 >$ `// Total Reduplication`
  else if $lev(w^1, w^2) = 1$ **then**
    return $< w^1\ w^2 >$ `// Partial Reduplication`
  else if w^1 *and* w^2 *are found in dictionary* **then**
    return $< w^1\ w^2 >$ `// MWEs other than Reduplication`
  else
    return w `// Non-MWE or whole word`

Table 3.1: Execution of the Algorithm for Structural Uniformity

Input (s)	**Index**	w^1	w^2	**MED**	w^1 & w^2 **in Dictionary**	**Action**
একচোখা ($\bar{e}kac\bar{o}kh\bar{a}$)	1	এ	কচোখা	5	No	Next Iteration
	2	এক	চোখা	4	Yes	Return এক-চোখা
খাওয়াদাওয়া ($kh\bar{a}w\bar{a}d\bar{a}w\bar{a}$)	1	খা	ওয়াদাওয়া	7	No	Next Iteration
	2	খাও	য়াদাওয়া	5	No	Next Iteration
	3	খাওয়া	দাওয়া	1	No	Return খাওয়া-দাওয়া

3.3.2 Removal of Semantically Insignificant Words

A key challenge in text preprocessing is the sheer volume of unique terms present in a corpus. Typically, millions of unique terms can be found in a large dataset, resulting in a vast feature space. This expansive feature space can negatively impact the performance and accuracy of text processing algorithms. Therefore, it is crucial to reduce the feature space by identifying and removing words that have little or no semantic significance. One category of such words is function words. Function words, such as conjunctions, prepositions, and articles, exist in a sentence to explain syntactic relationships among other words. Although they are essential for grammatical structure, they do not carry significant meaning on their own. In addition to function words, there are many content words in a corpus that may be insignificant in a particular context or domain.

3.3.2.1 Definitions and Notations

The definitions and notations bellow are utilized in the methodology pertain to the removal of insignificant words from a Bengali text corpus.

Representation of Documents and Corpus: Let $\kappa = \{w^1, w^2, \ldots, w^m\}$ be a set of m terms or words present in the corpus. Each document d_i, can then be represented as an m-vector $\mathbf{d}_i = (\omega_i^1, \omega_i^2, \ldots, \omega_i^m)$, where ω_i^j is the weight of term w^j in the document d_i. The document collection or corpus can be represented by a Term-Document Matrix D, where the columns of D are documents $d_1, d_2, \ldots, d_n$, and the rows of D are indexed by terms $w^1, w^2, \ldots, w^m$.

Each document d_i is assumed to be associated with one and only one class or category $C_i' \in \{C_1, C_2, \ldots, C_k\}$, where $k < n$ (e.g., Sports, Tourism, Literature, etc.). By the Pigeonhole Principle, multiple documents are contained within a single class. This principle ensures that while there may be many documents, they are categorized into a finite number of classes, resulting in some classes containing multiple documents.

Two weighing schemes have been employed for term weighting in the documents:

- **TF-IDF:** The weight ω_i^j is calculated using the Term Frequency-Inverse Document Frequency (TF-IDF) scheme:

$$\omega_i^j = \text{tf-idf}(w^j, d_i) = \text{tf}(w^j, d_i) \times \text{idf}(w^j) \tag{3.4}$$

where,

$$i = 1, 2, \ldots, n \quad \text{and} \quad j = 1, 2, \ldots, m.$$
$$\text{tf}(w^j, d_i) = \text{number of occurrences of the term } w^j \text{ in document } d_i.$$
$$\text{idf}(w^j) = \log\left(\frac{n}{\text{df}(w^j)}\right),$$
$$\text{df}(w^j) = \text{number of documents containing the term } w^j.$$

The term frequency $\text{tf}(w^j, d_i)$ measures how frequently a term w^j occurs in a document d_i. The inverse document frequency $\text{idf}(w^j)$ measures how much information the term provides, indicating if it is common or rare across all documents. The TF-IDF value increases proportionally with the number of times a term appears in the document but is offset by the frequency of the term in the corpus, which helps to adjust for the fact that some terms appear more frequently in general.

- **TF-IDF-ICF:** The weight ω_i^j is calculated using the Term Frequency-Inverse Document Frequency-Inverse Class Frequency (TF-IDF-ICF) scheme:

$$\omega_i^j = \text{tf-idf-icf}(w^j, d_i) = \text{tf-idf}(w^j, d_i) \times \text{icf}(w^j) \tag{3.5}$$

where,

$$i = 1, 2, \ldots, n \text{ and } j = 1, 2, \ldots, m.$$
$$\text{icf}(w^j) = \log\left(\frac{k}{\text{cf}(w^j)}\right),$$
$$\text{cf}(w^j) = \text{number of classes containing the term } w^j.$$

The inverse class frequency $\text{icf}(w^j)$ measures how much information the term provides based on its distribution across different classes. It is calculated as the logarithm of the ratio of the total number of classes to the number of classes containing the term w^j. This additional weighting helps to emphasize terms that are not only frequent in specific documents but also distributed across multiple classes, thereby enhancing the ability to distinguish between different classes.

Term Extraction Function: The function $T(D)$ is defined as the set of all terms present in a Term-Document Matrix D. Therefore, $|T(D)|$ represents the number of unique terms present in the Term-Document Matrix D. This function is essential for understanding the vocabulary size of the corpus and for subsequent processing steps, such as feature selection and dimensionality reduction.

3.3.2.2 Steps for Insignificant Words Removal

The removal of semantically insignificant words is an essential preprocessing step in many Natural Language Processing (NLP) pipelines. However, in the context of reduplication tagging, this step is not suitable to be applied beforehand, as it may inadvertently eliminate crucial components of reduplicated forms that carry linguistic or contextual significance. Instead, this step is more effectively applied after the completion of reduplication tagging. By deferring this process, the integrity of the reduplicated forms is preserved, ensuring accurate tagging and analysis. Once the tagging is completed, the removal of semantically insignificant words can significantly reduce the dimensionality of the data, thereby optimizing computational efficiency for downstream tasks such as text classification, information retrieval, and clustering. Below sequential steps ensures that essential linguistic structures are retained while still benefiting from the dimensionality reduction necessary for efficient processing.

Step I: Removal of Special Characters: In this step special characters are removed as described in Section 3.2.1.

Step II: Removal of Function Words: In a sentence, function words include pronouns, prepositions, conjunctions, qualifiers/intensifiers, determiners, auxiliary verbs, and interrogatives. In contrast, content words include nouns, verbs, adjectives, and adverbs. Utilizing the POS (Parts-of-Speech) tagged corpus, we retain words tagged as nouns, verbs, adjectives, and adverbs, while all other words are removed as function words.

Step III: Creation of Term-Document Matrix: After cleaning the corpus and retaining only the content words, the next step is to represent the documents in a structured format known as the Term-Document Matrix D. In this matrix:

- Each column represents a document from the corpus.
- Each row represents a term (content word) from the vocabulary.
- Each cell (i, j) in the matrix contains the weight of the term w^j in document d_i, which could be based weighting schemes TF-IDF (basic weighting scheme) and TF-IDF-ICF (advanced weighting scheme) as described in Section 3.3.2.1.

Step IV: Detection of Insignificant Content Words: The Term-Document Matrix D can be visualized as a collection of row vectors $w^1, w^2, \ldots, w^m$, where each vector represents the distribution of a term across all documents. So, every word w^j has a corresponding vector representation $\mathbf{w^j}$. Formally, each row vector w^j can be written as:

$$\mathbf{w^j} = (\omega_1^j, \omega_2^j, \ldots, \omega_m^j)^T$$

where ω_i^j is the weight of the term w^j in document d_i.

The algorithm identifies insignificant content words by examining these vectors. If a vector $\mathbf{w^j}$ is a zero vector (i.e., all elements are zero), it indicates that the word w^j appears in all the documents or classes. Such words are considered insignificant and can be removed from the matrix. This step is crucial for reducing the dimensionality of the matrix and focusing on terms that contribute meaningfully to the text analysis.

3.3.3 Results and Analysis

In this experiment, the "POS Tagged Bengali Corpus from TDIL", as described in Section 3.1, is utilized. Four data sets are prepared based on this corpus.

i. **Data Set 1 (D_1):** The given corpus contains 127,605 unique words. Steps I and Step II, described in the Section 3.3.2.2, removes 53,371 special characters and functional words. The remaining 74,234 words are present in 49 text documents of the corpus were represented as Data Set 1 (D_1). D_1 is used as the input for Step III as described in Section 3.3.2.2.

ii. **Data Set 2 (D_2):** If Step III as described in the Section 3.3.2.2, creates a Term-Document Matrix based on the tf-idf measure as specified in Section 3.3.2.1, it detects 33 content words as insignificant. These 33 insignificant content words were removed from D_1. The remaining 74,201 words present in 49 text documents of the corpus were represented as Data Set 2 (D_2). The following content words were removed:
If a term w^j was present in all documents, then $df(w^j) = n$, so $idf(w^j) = 0$ Therefore, $\omega_i^j = 0$ for $i = 1, 2, \ldots, n$ by Eq.3.4. Thus, w^j is detected as insignificant at Step IV as described in the Section 3.3.2.2, and removed from the corpus.

iii. **Data Set 3 (D_3):** If Step III as described in the Section 3.3.2.2, creates a Term-Document Matrix based on the tf-idf-icf measure as specified in Section 3.3.2.1, the algorithm detected 618 content words as insignificant. These 618 insignificant content words are removed from D_1. The remaining 73,616 words present in 49 text documents of the corpus are represented as Data Set 3 (D_3).
If a term w^j is present in all the classes, then $df(w^j) = n$, so $icf(w^j) = 0$. Therefore, $\omega_i^j = 0$ for $i = 1, 2, \ldots, n$ by Eq. 3.5. Thus, w^j is detected as insignificant at Step IV as described in the Section 3.3.2.2 and is removed from the corpus.

iv. **Data Set 4 (D_4):** By applying the *unify_mwe* (Algorithm 1) to Data Set 3 (D_3), the result will be an enhanced version, referred to as Data Set 4 (D_4). This transformation process involves the algorithm identifying and standardizing Multi-Word Expressions (MWEs) within the data set, ensuring structural uniformity. The *unify_mwe* algorithm processes each word in D_3, detecting and appropriately splitting MWEs based on the predefined rules. Consequently, D_4 represents a refined data set where MWEs are consistently formatted, potentially improving the accuracy and efficiency of subsequent text classification tasks performed on this data set.

Various supervised Machine Learning (ML) algorithms, such as Decision Tree (J48), PART, NaiveBayesMultinomial, NaiveBayes, and Optimized SVM (SMO), were executed on each data set. These ML algorithms were used for text document classification. Each data set was split into training and testing subsets using 10-fold cross-validation for these supervised ML algorithms. The accuracy of each algorithm was captured and compared among the data sets and presented in Table 3.2.

Table 3.2: Accuracy of ML Algorithms on Different Data Sets

ML Algorithm	D_1	D_2	D_3	D_4
Decision Tree (J48)	55.10%	55.10%	55.10%	59.10%
PART	44.90%	44.90%	44.90%	50.90%
NaiveBayesMultinomial	87.76%	89.80%	95.92%	96.92%
NaiveBayes	81.63%	81.63%	81.63%	86.63%
Optimized SVM (SMO)	79.59%	77.55%	77.55%	81.55%

From the results, it is evident that the accuracy remains the same for the three algorithms: Decision Tree (J48), PART, and NaiveBayes, even after the removal of 33 content words in Data Set 2 (D_2) and 618 content words in Data Set 3 (D_3) from Data Set 1 (D_1) by the proposed algorithm. However, the accuracy increases for NaiveBayesMultinomial with the removal of content words by the proposed algorithm. The accuracy decreases for SVM with the proposed algorithm for content word removal.

Also, by the definition of Data Sets provided above, 33 content words were removed from D_1 to create D_2. Therefore,

$$T(D_2) \subset T(D_1) \tag{3.6}$$

and

$$|T(D_1) - T(D_2)| = 33 \tag{3.7}$$

By Eq. 3.6 and Eq. 3.7, if all 33 elements present in the set of terms $T(\mathrm{D}_1)$ - $\mathrm{T}(D_2)$ are marked as insignificant and removed for further processing, there will be no significant change in accuracy in text classification.

Also, 618 content words were removed from D_1 to create D_3. Therefore,

$$T(D_3) \subset T(D_1) \tag{3.8}$$

and

$$|T(D_1) - T(D_3)| = 618 \tag{3.9}$$

By Eq. 3.8 and Eq. 3.9, if all 618 elements present in the set of terms $T(\mathrm{D}_1)$ - $\mathrm{T}(D_3)$ are marked as insignificant and removed for further processing, there will be no significant change in accuracy in text classification.

Again, if tf-idf$(w^j, d_i) = 0$, then tf-idf-icf$(w^j, d_i) = 0$ by Eq.3. It implies that the 33 content words removed from D_1 to create D_2 are also present in the 618 content words removed from D_1 to create D_3. Therefore,

$$T(D_3) \subset T(D_2) \tag{3.10}$$

and

$$|T(D_2) - T(D_3)| = 585 \tag{3.11}$$

Eq. 3.10 implies that the tf-idf-icf measure identifies the insignificant terms, which is a superset of those identified by the tf-idf measure. Eq. 3.11 implies that the tf-idf-icf measure performs better than tf-idf in terms of textual noise removal.

The *unify_mwe* technique, when applied to Data Set 3 (D_3), produces a new refined data set (D_4). This refinement further optimizes the accuracy of document categorization. As illustrated in Table 3.2, the application of *unify_mwe* results in an additional 5% improvement in classification accuracy. This significant enhancement underscores the effectiveness of the *unify_mwe* algorithm in reducing textual noise and identifying more precise Multi-Word Expressions (MWEs), leading to better performance of machine learning models in text classification tasks.

3.4 Conclusion

In conclusion, generic preprocessing and specialized preprocessing serve distinct yet complementary roles in Natural Language Processing (NLP) workflows. Generic preprocessing focuses on fundamental tasks such as text normalization, tokenization, and noise removal, ensuring that the raw text is clean, consistent, and ready for further analysis. These steps are broadly applicable across a wide range of tasks and form the foundation of any NLP pipeline. On the other hand, specialized preprocessing is tailored to address specific challenges unique to a particular task or domain. While generic preprocessing ensures a standardized and uniform input, specialized preprocessing enhances task-specific accuracy and performance. Together, these preprocessing strategies enable robust and efficient NLP systems by balancing generality and specificity.

CHAPTER 4

Identification of Reduplications

Reduplication, a linguistic phenomenon, is the repetition of a word or part of a word to convey a particular meaning or emphasize a concept. It is a common feature in many languages, including Bengali, and can serve various grammatical, semantic, and stylistic purposes. The identification of reduplications in texts is crucial for natural language processing (NLP) applications, such as text classification, machine translation, and sentiment analysis. By accurately identifying reduplicated forms, NLP systems can better understand the nuances and subtleties of language. This process involves distinguishing between full and partial reduplications, analyzing their structures, and understanding their contextual meanings. Effective identification of reduplications enhances the quality of language models and enables more precise interpretation of linguistic data, thus contributing significantly to the field of computational linguistics.

Contribution: This thesis presents methods for identifying and analyzing reduplications in Bengali using a combination of rule-based and machine learning approaches. A significant feature of the study is the integration of the Levenshtein Distance with an innovative technique called *Word Expansion* [7]. The development of a hybrid system, combining rule-based strategies with advanced machine learning algorithms, results in notable improvements in both detection accuracy and efficiency.

4.1 Rule Based Approach

4.1.1 Complete Reduplication Identification

Identification of complete reduplication (e.g., সকাল সকাল (*sakāl sakāl*: early morning), হাঃ হাঃ (*hāh hāh*: laughing loudly), is a trivial task from a computational perspective. Let $<w^1\ w^2>$ be the given bi-gram. If $w^1 = w^2$, then the bi-gram $<w^1\ w^2>$ can be considered a complete reduplication. In the context of Lavenstein distance, described in Equation 3.3, if $lev(w^1, w^2) = 0$, then $<w^1\ w^2>$ is a complete reduplication.

4.1.2 Partial Reduplication Identification

The study consists of two phases. In the initial phase, an algorithm utilizing Levenshtein distance is developed to detect partial reduplication in Bengali. In the subsequent phase, the limitations of the initial algorithm are addressed and rectified in the upgraded version.

4.1.2.1 Base Algorithm

The first phase of the study involved the design of an algorithm for identifying partial reduplication based on Levenshtein distance. The algorithm utilizes the concept of Levenshtein distance, which measures the similarity between two strings by calculating the minimum number of single-character edits required to transform one string into the other. A smaller Levenshtein distance indicates greater similarity between the words. The algorithm's description is as follows:

Algorithm 2: Base Algorithm for Identifying Partial Reduplication

Input: bi-gram $<w^1\ w^2>$
Output: Return TRUE if $<w^1\ w^2>$ is a partial reduplication, otherwise return FALSE
if $lev(w^1, w^2)$=1 **then**
  Return TRUE
Return FALSE

The Levenshtein distance serves as a measure of similarity between consecutive words in the Bengali text. When the distance is 1, it suggests the presence of partial reduplication due to minor variations between the words. Otherwise, the algorithm identifies the absence of partial reduplication.

When the bi-gram চাকরি বাকরি (*chākri bākri*: employment), if the letter 'চ' (*cha*) is substituted with the letter 'ব' (*ba*) in the first word চাকরি (*chākri*), it transforms

into বাকরি (*bākri*). As a result, the Levenshtein distance between চাকরি(*chākri*) and বাকরি (*bākri*) becomes 1. The algorithm used, the *Base Algorithm*, is capable of identifying this bi-gram চাকরি বাকরি (*chākri bākri*) as a partial reduplication due to the Levenshtein distance being equal to 1.

In another example, let's consider the input bi-gram আবোল তাবোল (*ābōl tabōl*: gibberish). Here, a minimum of two character edits are required to transform আবোল (*ābōl*) into তাবোল (*tabōl*). First, the letter 'আ' (*ā*) is substituted with the letter 'ত' (*ta*), and then the character 'া' (*ā-kār*) is inserted after 'ত' (*ta*). As a result, the Levenshtein distance between আবোল তাবোল (*ābōl tabōl*) becomes 2. So, the algorithm, *Base Algorithm*, is unable to identify this bi-gram আবোল তাবোল (*ābōl tabōl*) as a partial reduplication since the Levenshtein distance is equal to 2. This occurrence represents a false negative error for the *Base Algorithm*, as it fails to recognize the partial reduplication pattern in this particular case.

To address the false negative error, one possible approach is to use a Levenshtein distance of 2 to measure the similarity between words in step 1 of the *Base Algorithm*. However, this adjustment leads to a considerable increase in false positive instances, resulting in a decrease in the overall F1 Score from 80.00% to 75.91%, as shown in Figure 4.2. Consequently, setting higher values for the Levenshtein distance parameter in the *Base Algorithm* diminishes the overall accuracy of the algorithm.

4.1.2.2 Upgraded Algorithm

During the second phase of the study, extensive analysis was carried out to enhance the performance of the *Base Algorithm* by understanding the nature of false negative errors. The analysis revealed that for partial reduplications containing words with vowels as their first letters and their corresponding allographs attached to the first consonant of subsequent words, it requires two edits on the first character of the first word to make both words similar in the bi-gram. As a result, the Levenshtein distance between the constituents of the partial reduplication becomes equal to 2, leading the *Base Algorithm* to reject it.

For instance, consider the bi-gram আবোল তাবোল (*ābōl tabōl*), where the letter আ (*ā*) from the first word is used as the allograph of ত (*ta*) in the second word. First, the letter আ (*ā*) is substituted with the letter ত (*ta*), and then the character 'া' (*ā-kār*) is inserted after ত (*ta*) to transform আবোল (*ābōl*) into তাবোল (*tabōl*).

To address this problem without compromising the algorithm's performance, a noble technique called *Word Expansion* is employed on the input bi-gram before calculating the Levenshtein distance. This *Word Expansion* technique is elaborated in Equation 4.1.

Table 4.1: Mapping Table

Vowel Allograph(V_a)	Vowel Grapheme(V_g)
া	আ
ি	ই
ী	ঈ
ু	উ
ূ	ঊ
ৃ	ঋ
ে	এ
ৈ	ঐ
ো	ও
ৌ	ঔ

Word Expansion: It is a lexical operation applied to a Bengali word in which every **স্বরচিহ্ন** (*svaracinha*: vowel allograph) or **মাত্রা** (*mātrā*: vowel allograph) of the word is replaced by a corresponding **স্বরবর্ণ** (*svaravarna*: vowel grapheme). For example, this operation transforms the Bengali word **খাওয়া** (*khāwā*) into **খআাওয়আা**.

The *Word Expansion* operation on a string s containing Bengali alphabets, can be achieved using the following formula:

$$\text{we}(s) = \begin{cases} \text{f}(s_1) \cdot \text{we}(\text{tail}(s)) & \text{if } s_1 \in \text{V}_\text{a} \\ s_1 \cdot \text{we}(\text{tail}(s)) & \text{otherwise} \end{cases} \tag{4.1}$$

where tail(s) denotes the substrings obtained by removing the first character of s and s_1 represent the first character of strings s. Also, $s \cdot t$ represents the concatenation of string s and string t.

In this context, the transformation function $f : V_a \rightarrow V_g$, maps the set of Bengali vowel allographs (V_a) to the corresponding set of vowel graphemes (V_g) as shown in Table 4.1.

The description of the *Upgraded Algorithm* utilizing *Word Expansion* is provided in Algorithm 3:

Once again, let's consider the bi-gram **আবোল তাবোল** (*ābōl tabōl*) as the input for the *Upgraded Algorithm*. After applying *Word Expansion*, **আবোল তাবোল** is transformed into **আবওল তআবওল**. As a result, the Levenshtein distance between **আবওল** and **তআবওল** is now 1, since the deletion of the character **ত** (*ta*) from the second word makes both words similar. Consequently, the *Upgraded Algorithm* successfully identifies this bi-gram **আবোল তাবোল** (*ābōl tabōl*) as a partial reduplication. However, the *Base Algorithm* fails to recognize **আবোল তাবোল** (*ābōl tabōl*)

Algorithm 3: Upgraded Algorithm for Identifying Partial Reduplication

Input: bi-gram w^1 w^2
Output: Return TRUE if w^1 w^2 is a partial reduplication, otherwise return FALSE
$w_1 = we(w^1)$
$w_2 = we(w^2)$
if $lev(w_1, w_2) = 1$ **then**
  Return TRUE
Return FALSE

as a partial reduplication.

4.1.3 Flow Diagram

Let, a sentence be denoted as $s = \{w^1, w^2, \ldots, w^n\}$, where n represents the total number of words in the sentence s. Each pair of consecutive words $< w^k, w^{k+1} >$, where $k < n$, are treated as bi-grams.

Here, the bi-gram's constituent words are expanded using Equation 4.1, followed by the computation of the Levenshtein distance between them using Equation 3.3. When both words in the bi-gram exhibit a high degree of similarity, the Levenshtein distance is equal to 0, indicating the presence of complete reduplication. Conversely, a Levenshtein distance of 1 suggests the existence of partial reduplication.

The flowchart illustrating the process of bi-gram tokenization and reduplication identification can be referred to in Figure 4.1.

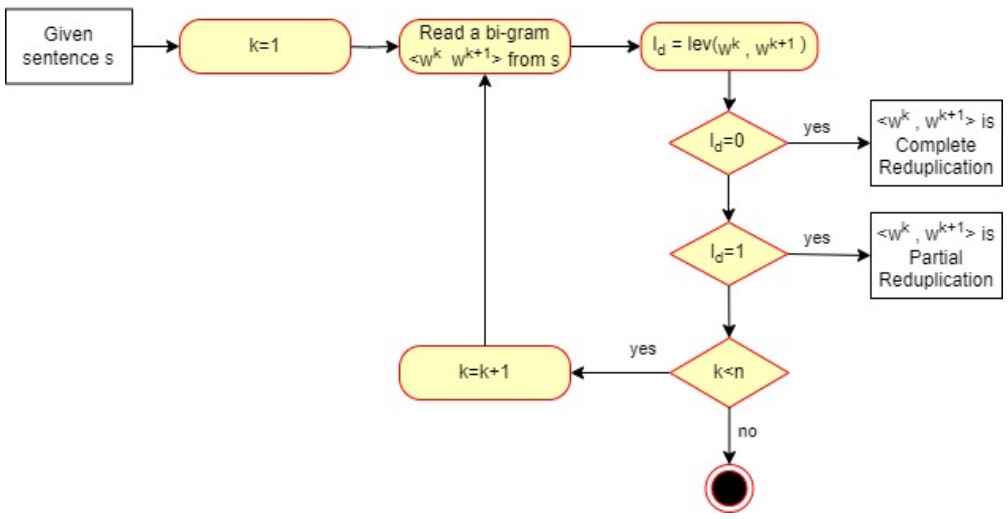

Figure 4.1: Identification of Complete and Partial Reduplication

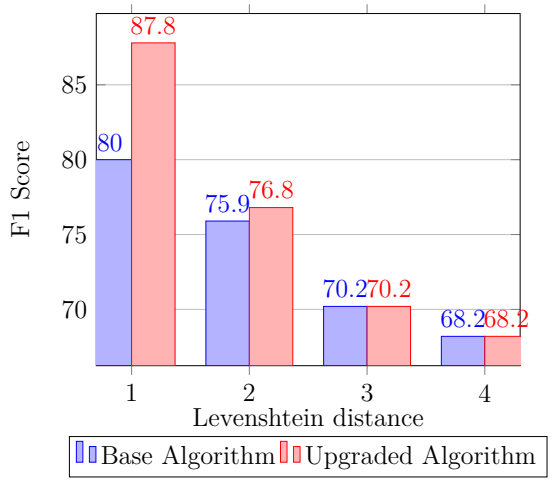

Figure 4.2: Comparison of Algorithm Performance

4.1.4 Optimal Levenshtein Distance Threshold

In the context of the study, the Levenshtein distance parameter plays a crucial role in measuring the similarity between two words. A higher Levenshtein distance indicates a larger dissimilarity between the words, as it represents the minimum number of single-character edits needed to transform one word into another. Therefore, with the increase of the Levenshtein distance parameter in both the *Base Algorithm* and the *Upgraded Algorithm*, the performance of both algorithms gradually reduces as presented in Figure 4.2. Increasing the Levenshtein distance threshold reduces false negative errors; however, it substantially increases false positive errors, leading to an overall decline in the F1-Score.

Interestingly, Figure 4.2 also indicates that both algorithms achieve their highest performance in terms of F1 Score when the Levenshtein distance is set to 1, which is considered the optimal threshold for identifying partial reduplication in the proposed approach.

4.1.5 Experiment Setup

Dataset: The experiment uses ”Reduplication List” and ”Bengali Corpus from TDIL”, as described in Section 3.1.

All sentences containing reduplications listed in the ”Reduplication List” dataset are extracted from the documents of the ”Bengali Corpus from TDIL”. Subsequently, these sentences are tokenized into bi-grams, considering two consecutive words. Each selected sentence for the experiment contains at least one bi-gram from the ”Reduplication List”. For example, let’s consider the Bengali sentence: **হোটেলের খাওয়া দাওয়া দুর্দান্ত ভালো** (*hoteler khāwā dāwā durdānto bhālo*: The food is excellent at the hotel). When tokenized by the bi-gram tokenizer, it yields the

following pairs: <হোটেলের খাওয়া>, <খাওয়া দাওয়া>, <দাওয়া দুর্দান্ত>, <দুর্দান্ত ভালো>. Here, <খাওয়া দাওয়া> represents the reduplication, while the other pairs do not exhibit reduplication.

Rule based Algorithms Execution: The proposed reduplication identification process is applied to each bi-gram according to the specifications outlined in the *Base Algorithm* and *Upgraded Algorithm*. A comparison of the performance of the proposed approaches is conducted against existing methods, including a Syntactic Rule-based approach [87] and a fuzzy rule-based system [88]. The entire execution process of the experiment is depicted in Figure 4.4.

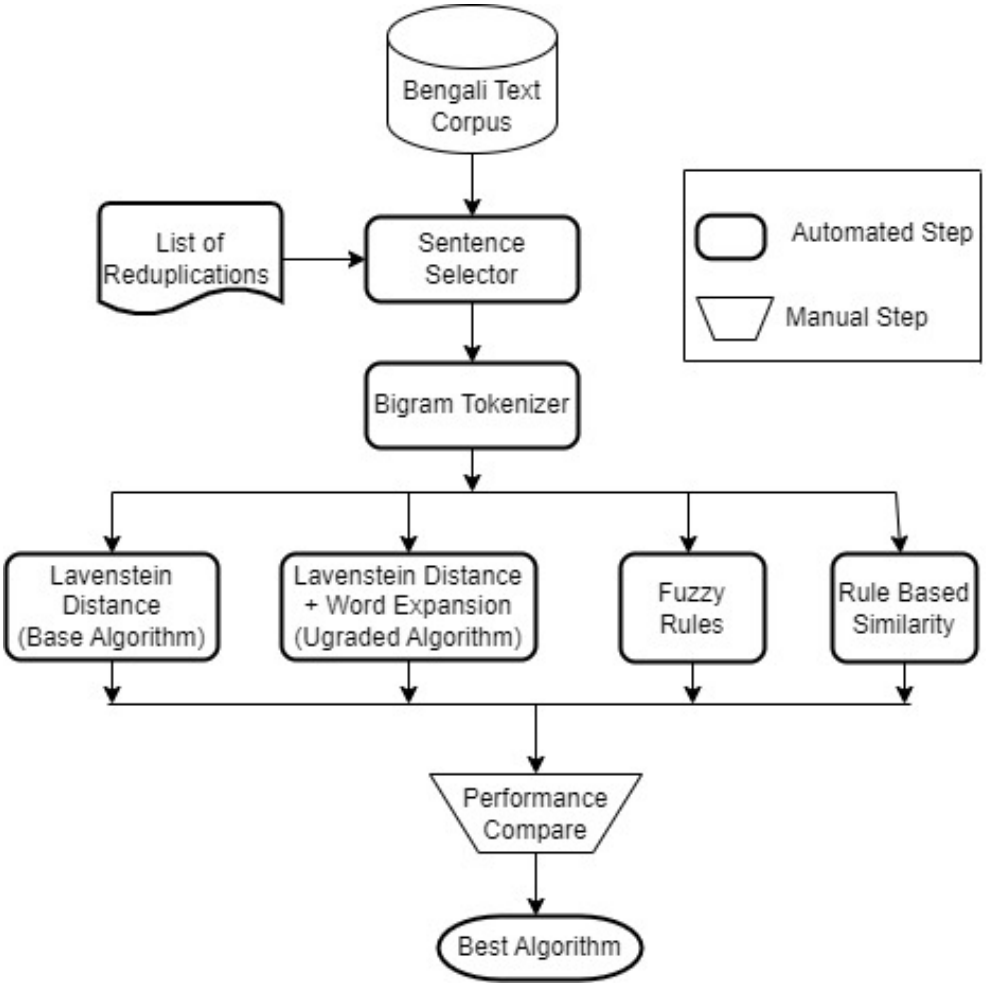

Figure 4.3: Experiment Execution Steps for Rule Based Approach

4.1.6 Results and Analysis

Comparative Study: Two approaches for reduplication identification have been proposed: the *Base Algorithm* which utilizes the Levenshtein distance measure, and the *Upgraded Algorithm*, which incorporates the Levenshtein distance measure

along with the *Word Expansion* strategy outlined in Equation 4.1. The performance of these approaches is compared with existing methods for reduplication identification based on the syntactic rules [87] and the fuzzy rules [88]. Since the proposed approach calculates the similarity between two given words, other statistics-based similarity scores are not considered for performance comparison in this context.

Identifying complete reduplication is relatively straightforward from a computational standpoint, and all four methodologies achieve 100% accuracy in this task. Therefore, their performance is evaluated based on their ability to identify Partial Reduplication, as detailed in Table 4.2.

Table 4.2: Performance Comparison for Rule Based Approach

System	Precision (%)	Recall (%)	F1-Score (%)
Levenshtein distance (*Base Algorithm*)	100.00	66.67	80.00
Levenshtein distance + *Word Expansion* (*Upgraded Algorithm*)	90.00	85.71	**87.80**
Syntactic Rules	87.50	66.67	75.68
Fuzzy Rules	78.26	85.71	81.82

In the domain of identifying reduplications in Bengali text, according to previous research, the syntactic rule [87] for identifying partial reduplications focuses solely on the first letter and its allograph in both words of a bi-gram. This limited approach poses significant challenges in detecting subsequent mismatches that may exist within the words of a partial reduplication. The fuzzy rule based system [88] has a provision to detect bi-gram as reduplication where there is low similarity between the components of the bi-gram. This feature results in more false positive errors. From Table 4.2, it is evident that the proposed approach, the *Upgraded Algorithm* that utilizes both Lavenstien distance and *Word Expansion* strategy, shows the highest F1 Score of 87.80%.

The key factor contributing to this enhanced performance lies in the utilization of the *Word Expansion* strategy in the *Upgraded Algorithm*. This innovative approach involves extending or augmenting the words under examination before calculating the Levenshtein distance. As a result, the algorithm becomes more robust in detecting partial reduplication patterns in Bengali words, leading to its superior performance.

Error Analysis: The proposed approach, which aims to identify partial reduplications in Bengali text, shows some limitations when two conditions co-occur within the same bi-gram. These conditions involve cases where the first letter of a

word in the bi-gram is a vowel, and its corresponding allograph is attached to the first consonant of the other word. Additionally, there may be a mismatch between the words in a later part of the bi-gram. For instance, let's consider the bi-gram উসখো খুসকো (*ushkhō kushkō*: disheveled). In this example, the vowel উ (*u*) is utilized as an allograph of the consonant খ (*kha*), and there is a mismatch at the third letter of the bi-gram. After applying the *Word Expansion* technique, the bi-gram transforms into উসখও খউসকও. Despite this transformation, the Levenshtein distance between the words remains 2, which leads to the *Upgraded Algorithm* failing to detect উসখো খুসকো (*ushkhō kushkō*) as a partial reduplication. Similarly, in the case of পরীক্ষা নিরীক্ষা (*parīkṣā nirīkṣā*: experiment), two conditions are observed: the change of the first consonant from প (*pa*) to ন (*na*), and the addition of vowel ঈ (*ī*). In this case as well, the proposed approach results in a false negative error. Some proper nouns may structurally resemble reduplications, although they are not. For instance, রাম রায় (Ram Roy) and কোকা কোলা (Coca Cola) are examples of such cases. These instances are classified as reduplications, leading to false positive errors for the proposed approach.

4.2 Machine Learning Based Approach

4.2.1 ML Features Selection

Statistical Features: The degree of association among the constituent words of a bi-gram is measured using statistical metrics. These metrics are computed using Table 4.3, which includes the bi-gram $<w^1w^2>$ represented by the Bengali words আবোল তাবোল (*ābōl tābōl*) as an example.

Table 4.3: Contingency Table

	$w^2 =$ তাবোল (*tābōl*)	$w^2 \neq$ তাবোল (*tābōl*)
$w^1 =$ আবোল(*ābōl*)	n_{11}	n_{12}
$w^1 \neq$ (*ābōl*)	n_{21}	n_{22}

Where,
n_{11} = number of times the bi-gram $<w^1\ w^2>$ appears in the corpus.
n_{12} = number of bi-grams containing w^1 in first position but not w^2 in the second position.
n_{21} = number of bi-grams containing w^2 in the second position but w^1 not in the first position.
n_{22} = number of bi-grams without both w^1 and w^2.
$N = n_{11} + n_{12} + n_{21} + n_{22}$ is the total number of bi-grams in the corpus.

The expected frequencies under the assumption of independence are given by:

$$e_{11} = \frac{(n_{11}+n_{12})\times(n_{11}+n_{21})}{N}$$
$$e_{12} = \frac{(n_{11}+n_{12})\times(n_{12}+n_{22})}{N}$$
$$e_{21} = \frac{(n_{21}+n_{22})\times(n_{11}+n_{21})}{N}$$
$$e_{22} = \frac{(n_{21}+n_{22})\times(n_{12}+n_{22})}{N}$$

The statistical measures have been considered, as specified below [89]:

(a) simple-II: $2 \times (n_{11} \log(\frac{n_{11}}{e_{11}}) - (n_{11} - e_{11}))$

(b) t-Score: $\frac{n_{11}-e_{11}}{\sqrt{n_{11}}}$

(c) z-Score: $\frac{n_{11}-e_{11}}{\sqrt{e_{11}}}$

(d) MI: $\log(\frac{n_{11}}{e_{11}})$

(e) MI^2: $\log(\frac{n_{11}^2}{e_{11}})$

(f) Dice: $\frac{2\times n_{11}}{2\times n_{11}+n_{12}+n_{21}}$

(g) Log-Likelihood: $2 \times (n_{11} \log(\frac{n_{11}}{e_{11}}) + n_{12} \log(\frac{n_{12}}{e_{12}}) + n_{21} \log(\frac{n_{21}}{e_{21}}) + n_{22} \log(\frac{n_{22}}{e_{22}}))$

(h) Chi-square (χ^2): $\frac{(n_{11}-e_{11})^2}{e_{11}} + \frac{(n_{12}-e_{12})^2}{e_{12}} + \frac{(n_{21}-e_{21})^2}{e_{21}} + \frac{(n_{22}-e_{22})^2}{e_{22}}$

(i) NPMI: $\frac{MI}{-\log(\frac{n_{11}+n_{21}}{N})}$

(j) ΔP_1: $\frac{n_{11}}{n_{11}+n_{12}} - \frac{n_{21}}{n_{21}+n_{22}}$

(k) ΔP_2: $\frac{n_{11}}{n_{11}+n_{21}} - \frac{n_{12}}{n_{12}+n_{22}}$

Syntactical Features: Phonological patterns [90] play a significant role in shaping the structure of Bengali reduplicated forms. These patterns refer to the specific ways in which sounds or phonemes are organized and repeated within words, contributing to the distinct phonological identity of reduplications in the Bengali language. Understanding these patterns is crucial for analyzing the formation and use of reduplicated expressions, as they directly influence how these expressions are perceived and understood by speakers of the language. These phonological patterns are described in Table 4.4 which provides a comprehensive overview of the different ways in which sounds are arranged and repeated in Bengali reduplication. For instance, a word composed of a consonant (C) followed by a vowel (V) will create a reduplication if repeated, as indicated in SL. No.1 of the Table 4.4.

Each pattern operates as a binary feature, indicating the presence or absence of specific phonological characteristics within the reduplicated forms.

Levenshtein distance: The Levenshtein distance, as described in Section 3.3.1.1, is a metric used to measure the similarity between two strings of a bi-gram.

Table 4.4: Phonological Patterns of Reduplication

SL No.	Pattern	Example
1	$CV_{[a/e/i/o/u]} : CV_{[a/e/i/o/u]}$	টা টা (*ṭā ṭā*)
2	$CV_{[u]}CV_{[u]}C : CV_{[a]}CV_{[u]}C$	পুটুস পাটুস (*puṭus pāṭus*)
3	$CV_{[a]}CV_{[u]}C : CV_{[u]}CV_{[u]}C$	টাপুর টুপুর (*ṭāpur ṭupur*)
4	$CV_{[u]}CV_{[o]} : CV_{[a]}CV_{[a]}$	কুচো কাঁচা (*kuco kāñcā*)
5	$CV_{[u]}CCV_{[o]} : CV_{[a]}CCV_{[a]}$	টুকরো টাকরা (*ṭukro ṭākrā*)
6	$CV_{[o]}CV_{[a]} : CV_{[u]}CV_{[i]}$	মোটা মুটি (*moṭā muti*)
7	$CV_{[a]}CV_{[a]} : CV_{[a]}CV_{[i]}$	কাচা কাচি (*kācā kāci*)
8	$CCV_{[a]} : CCV_{[i]}$	করা করি (*karā kari*)
9	$CV_{[a]}CV_{[a]} : CV_{[u]}CV_{[i]}$	কাটা কুটি (*kāṭā kuṭi*)
10	$CV_{[u]}CV_{[o]} : CV_{[u]}CV_{[i]}$	গুতো গুতি (*guṭo guṭi*)
11	$CV_{[e]}CV_{[a]} : CV_{[e]}CV_{[i]}$	দেখা দেখি (*dekhā dekhi*)
12	$CV_{[a]}V_{[i]} : CV_{[a]}V_{[i]}$	কাই কাই (*kāi kāi*)
13	$CV_{[a]}V_{[u]} : CV_{[a]}V_{[u]}$	কাউ কাউ (*kāu kāu*)
14	$CV_{[i]}CV_{[i]} : CV_{[i]}CV_{[i]}$	টিপি টিপি (*ṭipi ṭipi*)
15	$CV_{[i]}C : CV_{[i]}C$	কিট কিট (*kiṭ kiṭ*)
16	$CVC : CVCV_{[i]}$	কাট কাটি (*kaṭ kāṭi*)
17	$CC : CCV_{[a]}CV_{[i]}$	কল কলানি (*kal kalāni*)
18	$CC : CCV_{[e]}$	কর করে (*kar kare*)
19	$CV_{[i]}C : CV_{[i]}CV_{[e]}$	লিক লিকে (*lik like*)
20	$CV_{[u]}C : CV_{[u]}CV_{[e]}$	কুট কুটে (*kuṭ kuṭe*)

4.2.2 Experiment Setup

Dataset: This experiment also uses the same dataset detailed in Section 4.1.5.

Feature Selection: For each bi-gram analyzed, a comprehensive set of statistical and syntactical features is computed, as detailed in Section 4.2.1. These features encompass twelve statistical measures and twenty-one syntactical characteristics, resulting in a structured dataset with a 33-dimensional feature vector for each entry.

Class Label: Each entry in this dataset is annotated with a binary class label, where '1' signifies the presence of reduplicated forms and '0' represents other instances. This annotated dataset forms the basis for training and testing various Machine Learning algorithms.

ML Algorithms Execution: In this experiment, a comparative study of performance is conducted using four machine learning algorithms: Naive Bayes, Support

Vector Machine (SVM), Decision Tree (J48), and Random Forest. These algorithms were chosen to assess their effectiveness in addressing the identification of reduplicated forms in Bengali. The entire execution process of the experiment is depicted in Figure 4.4.

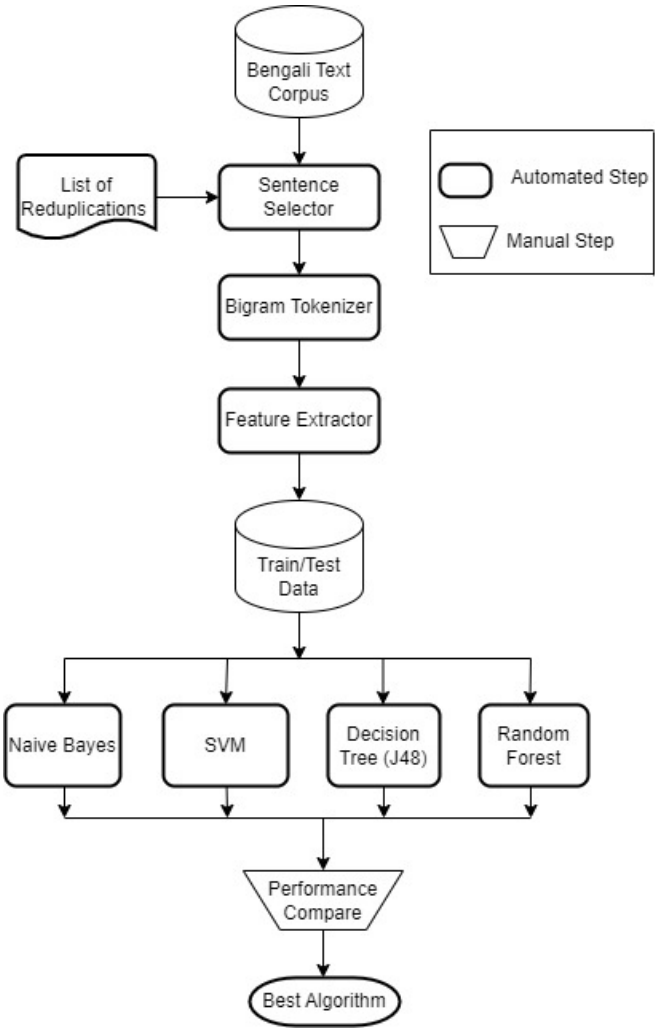

Figure 4.4: Experiment Execution Steps for ML Based Approach

4.2.3 Results and Analysis

Comparative Study: The machine learning tool Weka[1] was utilized for the experiment. The random forest classifier, which is included under the "Classifier/trees" panel of the Weka workbench, was employed. The random forest classifier of the Weka suite was run with the default values of its parameters for this work. One of the important parameters is the number of trees in the forest, which was set to its default value of 10. To estimate the overall accuracy of the proposed reduplication identification system, 10-fold cross-validation was performed.

[1]https://ml.cms.waikato.ac.nz/index.html

Identifying complete reduplication instances (e.g., সকাল সকাল (*sakāl sakāl*: early morning), হাঃ হাঃ (*hāh hāh*: laughing loudly) is straightforward computationally, and all four methodologies achieve 100% accuracy in this task. Let $<w^1$ $w^2>$ represent the given bi-gram. If w^1 equals w^2, then the bi-gram $<w^1$ $w^2>$ can be classified as a complete reduplication. In the context of the Levenshtein distance in Equation 3.3, if $lev(w^1, w^2) = 0$, then $<w^1$ $w^2>$ is identified as a complete reduplication. Hence, their performance is assessed primarily based on their capability to detect partial reduplication.

Table 4.5: Performance Comparison for ML Based Approach

System	Precision (%)	Recall (%)	F1-Score (%)
Naive Bayes	88.24	71.42	78.94
Support Vector Machine	89.47	80.95	85.00
Decision Tree	88.89	76.19	82.05
Random Forest	90.90	95.24	93.02

The estimated accuracy of this system is shown in Table 4.5. Also depicted in table is the comparison of the performance of various ML algorithms with respect to reduplication identification. It is observed that a random forest-based system gives an average F-measure of 93.02%, which is the highest among four ML algorithms for the reduplication recognition task.

Error Analysis: In some cases, certain proper nouns may exhibit structural similarities to reduplications, even though they are not actual instances of reduplication. For example, names like রাম রায় (Ram Roy) and কোকা কোলা (Coca Cola) resemble reduplicated forms, but they are not. However, these instances might erroneously be classified as reduplications by the proposed approach, resulting in false positive errors.

4.3 Conclusion

This chapter focuses on an algorithm for detecting partial reduplication in written Bengali texts. The algorithm uses minimum edit distance to measure word similarity by calculating the minimum single-character edits needed to transform one word into another. To enhance accuracy, a new lexical operation called *Word Expansion* is introduced. This deepens understanding of word structure, improving the detection of partial reduplication. The results demonstrate that the *Upgraded Algorithm*, incorporating Levenshtein distance and *Word Expansion*, outperforms other systems, yielding impressive performance in reduplication identification.

This chapter also focuses the utilization of Machine Learning algorithms to identify partial reduplication in written Bengali texts. These algorithms leverage a wide range of statistical and syntactical features. Among the classifiers employed, the Random Forest demonstrates the most promising performance.

CHAPTER 5

Generation of Reduplications

The generation of reduplications is a fascinating linguistic process that involves the repetition of a word or part of a word to create new expressions or to add emphasis, variety, or intensity to language. This phenomenon is found in many languages across the world and serves various purposes, such as expressing plurality, intensifying meaning, or creating rhythm and sound patterns in speech. Reduplication can take many forms, including the repetition of entire words (full reduplication) or just parts of words (partial reduplication). The process is not just a simple repetition; it often follows specific linguistic rules and patterns unique to each language. Reduplication enriches the expressive capacity of a language, allowing speakers to convey subtle shades of meaning and emotion. Understanding the generation of reduplications provides valuable insights into the flexibility and creativity inherent in human language.

Contribution: This research explores the generative aspects of reduplications, examining their creation and structural patterns. Finite State Transducers (FSTs) are utilized as computational models to simulate the generation and identification of reduplicated forms [8].

5.1 Machine Model for Reduplication

Beesley et al. [69] argue that the complexity of phonological and morphological processes across languages can be systematically captured through the use of regular expressions. These expressions, in turn, can be compiled into finite-state transducers, which are powerful computational tools capable of modeling and processing these linguistic phenomena. This assertion highlights the potential for formalizing and automating the analysis of language patterns, making it

possible to create precise and efficient language processing tools.

In the context of Bengali, Dash [90] extends this idea by identifying specific phonological patterns and regular expressions that account for the formation of reduplications—a widespread and distinctive feature of the language. These patterns serve as theoretical frameworks that explain how different types of reduplications are generated in Bengali. Some of the more frequent and easily recognizable patterns are presented in Table 4.4.

This combined approach, leveraging the principles of finite-state transducers and the detailed study of language-specific phonological patterns, offers a robust methodology for understanding and generating reduplications in Bengali. It underscores the intersection of computational linguistics and traditional linguistic theory, demonstrating how advanced computational models can be grounded in the understanding of a language's phonological and morphological rules.

2-way Finite-state Transducer (FST) A 2-way FST is a machine that processes an input string from an alphabet Σ and produces an output string from an alphabet Γ. The input string is flanked with the start ($\langle$) and end ($\rangle$) symbols, which are not present in Σ. Starting in an initial state q_0, the FST reads an input symbol from the input tape, changes its internal state, writes a corresponding output string to the output tape, and moves its read head left or right, while the writing head only moves forward. This process continues until the read head consumes the entire input string. If the FST reaches an accepting state after consuming the input string, it signifies that a valid output string has been generated corresponding to the input string. The operational model of 2-way FST is shown in Figure 5.1.

Below is the mathematical definition of 2-way deterministic FST based on definitions from Filiot et al. [71] and Shallit [91].

$$M = (Q, \Sigma_{\langle\rangle}, \Gamma, \delta, q_0, F)$$

where:

- Q is a finite set of states,
- $\Sigma_{\langle\rangle} = \Sigma \cup \{\langle, \rangle\}$ where Σ is the input Bengali alphabet,
- $\Gamma = \Sigma \cup \{-\}$ is the output alphabet,
- δ is the transition function,
- q_0 is the initial state,
- F is the set of final states.

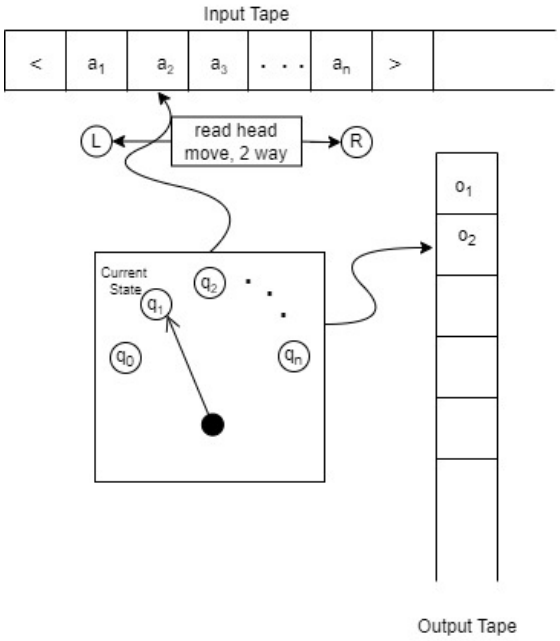

Figure 5.1: Working Model of 2-Way FST

The transition function δ maps a current state and input symbol to the next state, output string, and direction of movement (left or right) on the input tape.

$$\delta : Q \times \Sigma \to Q \times \Gamma^* \times \{L, R\}$$

A configuration of the 2-way deterministic FST is represented as (q, w, i, v), where:

- q is the current state ($q \in Q$),
- w is the content of the input tape ($w \in \Sigma^*$),
- i is the position of the read/write head on the input tape ($i \in \Sigma$),
- v is the content of the output tape ($v \in \Gamma^*$).

The read head of a 2-way FST can move both forward and backward, making it well-suited for modeling complex patterns like reduplication. In contrast, 1-way FSTs process input only in a single forward direction, limiting their ability to effectively handle such patterns.

5.2 2-Way FST for Complete Reduplication

This section provides a detailed explanation of the methodology involved in representing Bengali complete reduplication through the use of a finite-state trans-

ducer. Complete reduplication, a morphological process with broad applications, is observed across diverse word classes in the Bengali language. Reduplicated forms are not only present in nouns but also in pronouns, verbs, modifiers (both nominal and verbal), numerals, and interjections. This demonstrates the extensive and versatile nature of complete reduplication as a productive morphological phenomenon.

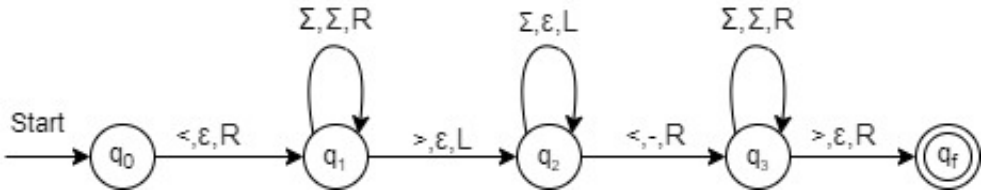

Figure 5.2: 2-Way FST for Complete Reduplication in Bengali

Figure 5.2 illustrates the design of a 2-way FST for the complete reduplication generation process in Bengali. The input tape of the 2-way FST initially contains a Bengali word enclosed by the start (<) and end (>) markers. The symbol Σ represents any element in the alphabet, excluding the start and end markers $\{<, >\}$. It executes the following steps to generate the corresponding reduplicated form on the output tape.

i. **Transition q_0 to q_1**: Reads < and moves right without writing anything (ϵ).

ii. **Transition q_1 to itself**: Scans the input tape from left to right to generate the base element.

iii. **Transition q_1 to q_2**: Reads > and moves the head left.

iv. **Transition q_2 to itself**: Moves the head back to the start of the input tape until < is encountered.

v. **Transition q_2 to q_3**: Adds a boundary symbol, represented by a hyphen ('-'), to differentiate between the two words of the reduplication.

vi. **Transition q_3 to itself**: Reads the input tape once more from left to right until the reduplicant element is generated.

vii. **Transition q_3 to q_f**: Ends the process.

Table 5.1 illustrates the configuration and state transitions of the FST responsible for generating the reduplication হাসি-হাসি (*hāsi-hāsi*: smiling). The process begins with হাসি(*"hāsi"*) as the base word on the input tape. The table details each step of the FST, showing how it processes the input to produce the reduplicated form. Specifically, it outlines how the FST reads the base word, navigates back to the start, inserts a boundary symbol, and finally generates the reduplicant to create the complete output হাসি-হাসি (*hāsi-hāsi*).

Table 5.1: Execution Steps of FST for Generation of *hāsi-hāsi*

Step	*Head Position*	*Present State*	*Head Movement*	*Next State*	*Output*
1	<u>≤</u>hāsi>	q_0	R	q_1	ϵ
2	<<u>h</u>āsi>	q_1	R	q_1	h
3	<h<u>ā</u>si>	q_1	R	q_1	hā
4	<hā<u>s</u>i>	q_1	R	q_1	hās
5	<hās<u>i</u>>	q_1	R	q_1	hāsi
6	<hāsi<u>≥</u>	q_1	L	q_2	hāsi
7	<hās<u>i</u>>	q_2	L	q_2	hāsi
8	<hā<u>s</u>i>	q_2	L	q_2	hāsi
9	<h<u>ā</u>si>	q_2	L	q_2	hāsi
10	<<u>h</u>āsi>	q_2	L	q_2	hāsi
11	<u>≤</u>hāsi>	q_2	R	q_3	hāsi-
12	<<u>h</u>āsi>	q_3	R	q_3	hāsi-h
13	<h<u>ā</u>si>	q_3	R	q_3	hāsi-hā
14	<hā<u>s</u>i>	q_3	R	q_3	hāsi-hās
15	<hās<u>i</u>>	q_3	R	q_3	hāsi-hāsi
16	<hāsi<u>≥</u>	q_3	R	q_f	hāsi-hāsi

In this example, the process is broken down into several key stages. Steps 1-6 involve generating the base word, where the input tape is read and processed to form the initial word. Following this, steps 7-10 move the head of the tape back to the starting position, ensuring that the system is ready to handle the next phase of the operation. Step 11 introduces the boundary symbol, represented by a hyphen ('-'), which serves to separate the base word from its reduplicated form. Finally, steps 12-16 focus on producing the reduplicant word on the output tape, effectively duplicating the initial base word to complete the reduplication process.

5.3 2-Way FST for Partial Reduplication

This section explains the modeling of partial reduplication in Bengali using a set of finite-state transducers. Partial reduplication is a prevalent linguistic phenomenon found in approximately 75% of languages worldwide, as reported by Rubino [92]. Moreover, it constitutes a common reduplication process within the Bengali language. The process of partial reduplication in Bengali involves the duplication of a base element, which is subsequently affixed, suffixed, or even infix-fixed to the original base. This operation introduces phonological alterations in the copied element, occurring at various positions, be they initial, medial, or final. The phonological modifications may encompass alterations, insertions, or

deletions of consonants or vowels, or even entire syllables within the duplicated element.

In Bengali, Dash [90] has identified and documented various patterns of partial reduplication. These patterns exhibit a high level of productivity, suggesting their frequent occurrence and application within the language. One specific example of a partial reduplication pattern is illustrated in Figure 5.3 through a 2-way FST. This particular pattern, denoted as "CV[a]CV[u]C" in its initial input, transforms into the reduplicated pattern "CV[a]CV[u]C CV[u]CV[u]C" as represented by the FST, denoted as "CV[a]CV[u]C" → "CV[a]CV[u]C CV[u]CV[u]C". The FST serves as a visual representation of the computational process involved in applying this specific reduplication pattern, providing a valuable tool for understanding and implementing such linguistic phenomena.

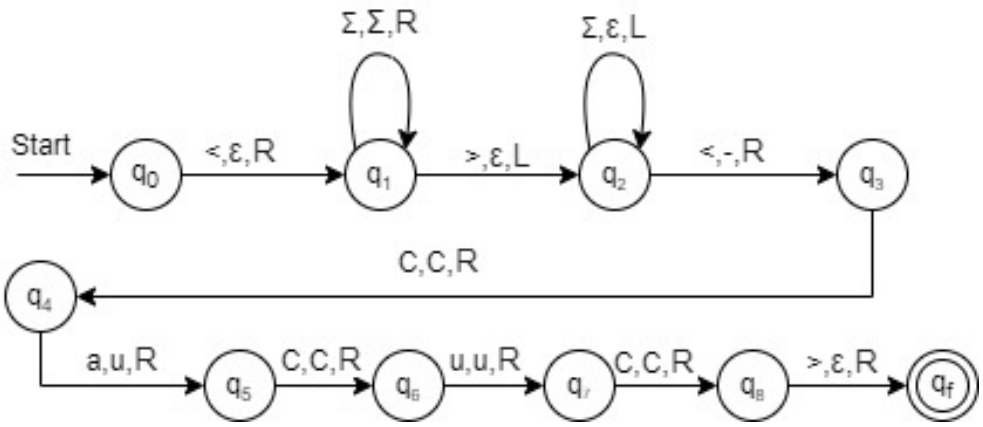

Figure 5.3: 2-Way FST for Partial Reduplication in Bengali

An example of derivation for *"tāpur"* → *"tāpur-tupur"* is provided using the 2-way FST in Table 5.3. The computation for the input word *"tāpur"* is described step by step through the utilization of the 2-way FST depicted in Figure 5.3. The modeling of each partial reduplication can be achieved through 2-way FSTs once the reduplicant pattern is known.

5.4 Reduplication tagging with FST

The process of identifying reduplication involves the creation of 2-way FSTs for various reduplication patterns [90]. Subsequently, each word in a given sentence is subjected to these FSTs. The objective is to generate reduplication expressions by applying the patterns encoded in the FSTs. If both words of the generated bigram are found in the provided sentence, the bigram is tagged as a reduplication. The algorithm for identifying reduplication using finite-state transducers (FST) is outlined in Algorithm 4.

Table 5.2: Execution Steps of FST for Generation of *tāpur-tupur*

Step	*Head Position*	*Present State*	*Head Movement*	*Next State*	*Output*
1	≤tāpur>	q_0	R	q_1	ϵ
2	<t̲āpur>	q_1	R	q_1	t
3	<tā̲pur>	q_1	R	q_1	tā
4	<tāp̲ur>	q_1	R	q_1	tāp
5	<tāpu̲r>	q_1	R	q_1	tāpu
6	<tāpur̲>	q_1	R	q_1	tāpur
7	<tāpur≥	q_1	L	q_2	tāpur
8	<tāpur̲>	q_2	L	q_2	tāpur
9	<tāpu̲r>	q_2	L	q_2	tāpur
10	<tāp̲ur>	q_2	L	q_2	tāpur
11	<tā̲pur>	q_2	L	q_2	tāpur
12	<t̲āpur>	q_2	L	q_2	tāpur
13	≤tāpur>	q_2	R	q_3	tāpur-
14	<t̲āpur>	q_3	R	q_4	tāpur-t
15	<tā̲pur>	q_4	R	q_5	tāpur-tu
16	<tāp̲ur>	q_5	R	q_6	tāpur-tup
17	<tāpu̲r>	q_6	R	q_7	tāpur-tupu
18	<tāpur̲>	q_7	R	q_8	tāpur-tupur
19	<tāpur≥	q_8	R	q_f	tāpur-tupur

In the context of the sentence **টাপুর টুপুর বৃষ্টি পড়ে** (*tāpur tupur brishti pore*: Raindrops are falling gently), when the term *"tapur"* undergoes processing through the FST illustrated in Figure 5.3, it produces *"tāpur-tupur"* on the output tape. Since both *"tāpur"* and *"tupur"* are identified within the original sentence, the expression *"tapur tupur"* is consequently labeled as reduplication in the provided sentence.

5.5 Results and Analysis

This experiment utilizes the "Reduplication List" as the primary dataset, along with the "Bengali Corpus from TDIL" and the "Bengali Corpus from Wikipedia," as detailed in Section 3.1. Sentences containing reduplications from the "Reduplication List" dataset are extracted from these two text corpora. Subsequently, the proposed algorithm is applied to both the TDIL and Wiki datasets, and the results obtained are presented in the Table 5.3.

The algorithm demonstrates strong performance in identifying reduplication instances within the TDIL dataset, achieving a precision of 89.00%. This indicates its high accuracy in distinguishing true positive reduplications from false

Algorithm 4: Algorithm for Identifying Reduplication

Input: Sentence S, Set of FSTs F
Output: Reduplication tagged sentence of S denoted by S_R
$S_R \leftarrow S$
Tokenize S as $\{w^1, w^2, \ldots, w^n\}$
for *each* w^i **do**
  for *each* $F_j \in F$ **do**
    Input tape of $F_j \leftarrow w^i$
    Run F_j
    if F_j *does not HALT* **then**
      Bigram $w^{i1}w^{i2} \leftarrow$ Output tape of F_j
      if $w^{i1} \in S$ *and* $w^{i2} \in S$ **then**
        $S_R \leftarrow$ Tag bigram $<w^{i1}w^{i2}>$ as Reduplication in S_R
return S_R

Table 5.3: Performance of FST Based Approach

Data Set	Precision	Recall	F1-Score (%)
TDIL	89.00	86.67	87.81
Wiki	91.04	85.36	88.11

positives. With a recall rate of 86.67%, the algorithm captures a significant portion of actual reduplications, though there is slight room for improvement. The F1-Score of 87.81% balances these metrics, highlighting the algorithm's reliable and well-rounded performance in practical applications for this dataset.

In the Wiki dataset, the algorithm shows improved precision at 91.04%, reflecting its enhanced accuracy in identifying reduplications. The recall rate is slightly lower at 85.36%, suggesting some challenges in recognizing all instances within this diverse dataset. However, the F1-Score of 88.11% underscores the algorithm's overall strong performance, effectively balancing high precision with competent recall. This consistency across different datasets indicates the algorithm's robustness and reliability in various contexts.

The FST-based rules exhibit strong performance across datasets, achieving an F1-Score of 88.11% for Wiki Data and 87.81% for the TDIL Corpus, as shown in Table 5.3. These rules outperform other rule-based methods for reduplication identification, as detailed in Table 4.2. This consistent performance across datasets underscores the algorithm's adaptability and effectiveness, making it a valuable asset for linguistic analysis and NLP tasks involving Bengali reduplication.

Error Analysis: Proper nouns can sometimes exhibit structural features that mimic the appearance of reduplications, leading to potential challenges in accu-

rate linguistic classification. For instance, names such as রাম রায় (Ram Roy) and কোকা কোলা (Coca Cola) bear a superficial resemblance to reduplicative patterns. Classifying such instances as reduplications can introduce false positives, thereby compromising the accuracy and reliability of the proposed approach.

5.6 Conclusion

The study aims to develop a computational model that can accurately represent both partial and complete reduplication in the Bengali language using 2-way FSTs. It also introduces a novel approach employing those 2-way FSTs to detect instances of reduplication in the Bengali language. The performance of this system is rigorously assessed using two distinct datasets: the list of Multi-Word Expressions (MWEs) published by TDIL, MeitY - Government of India, and a Bengali corpus obtained from Bengali Wikipedia. The proposed algorithm achieves impressive results, demonstrating a noteworthy F1-Score of 88.11% for the Wiki Data dataset and a corresponding F1-Score of 87.81% for the TDIL Corpus. These scores indicate the algorithm's proficiency in accurately identifying reduplications within Bengali linguistic data.

CHAPTER 6

Sense Tagging of Reduplications

Sense tagging of reduplications involves assigning semantic labels to repeated words or phrases within a text. Reduplication, a prevalent linguistic feature across many languages, entails the repetition of a word or part of a word to create a new expression, often with altered or intensified meaning. This feature serves various functions, such as emphasizing, indicating plurality, or conveying subtle meanings. However, interpreting the semantics of reduplications can be complex due to their context-dependent nature. Sense tagging in this context aims to clarify these meanings, enhancing our understanding of the roles and significance of reduplicated forms in language. This systematic categorization aids in improving natural language processing systems, leading to more accurate text analysis and better language comprehension.

Contribution: This study examines the semantic layer of Bengali reduplications by analyzing their contextual meanings using BERT models for accurate interpretation. The research culminates in a dynamic repository of Bengali reduplications and their meanings, designed to evolve with new data.

6.1 Definition of Operations

BERT Embedding: BERT embedding involves creating rich, contextual representations of words or tokens using the BERT (Bidirectional Encoder Representations from Transformers) model. Unlike traditional word embeddings that assign a fixed vector to each word regardless of its context, BERT, a deep learning model developed by Google, is designed to understand and represent words in relation to all other words in a sentence, rather than just in isolation or based on adjacent words. This approach allows for a more sophisticated and accurate representation

of language, capturing the subtleties of word usage that depend on context.

When a sentence is fed into BERT, the model first tokenizes the sentence, breaking it down into smaller units called tokens, which could be words or sub-words. BERT then processes these tokens through multiple layers of transformers, capturing the complex relationships and meanings within the entire sentence.

Each token is assigned a numerical vector that reflects its meaning within the specific context of the sentence. For example, in the case of Bengali text, BERT would generate a 768-dimensional vector for each token, representing its unique contextual meaning.

Consider a sentence, denoted as $s = \{w^1, w^2, \ldots, w^n\}$, which serves as the input to the pretrained BERT model. When this sentence is tokenized using the model's tokenizer, it may potentially divide a single word into several tokens. Let's assume that a word w^i is segmented into a set of BERT tokens, $t^i_1, t^i_2, \ldots, t^i_{m_i}$. Now, for each BERT token t^i_j, where $i \leq n$ and $j \leq m_i$, the model generates a numerical vector, $e^i_j = [x^i_{j1}, x^i_{j2}, \ldots, x^i_{jp}]$, where $p = 768$ when Multilingual BERT [76] or Bengali BERT [77] is employed.

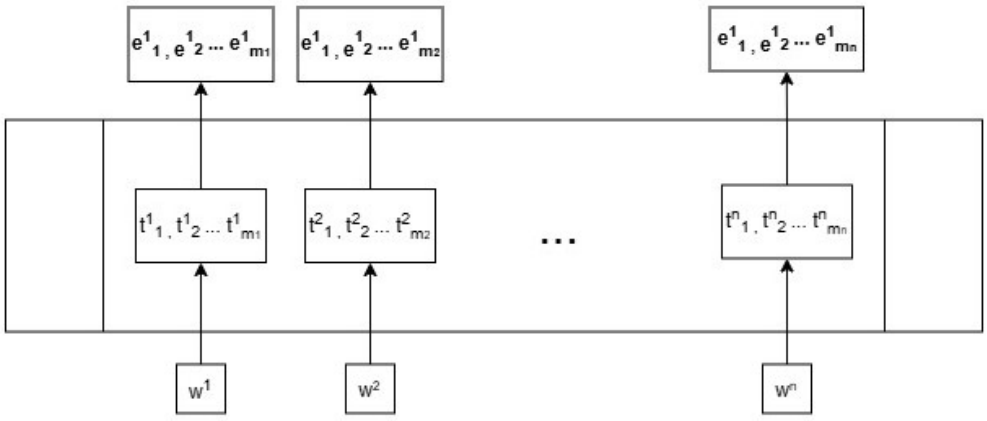

Figure 6.1: Pretrained BERT Embedding

The representation of a word w^i as an embedding or vector is symbolized as $\mathbf{e^i}$, and it is derived through the following equation:

$$\mathbf{e^i} = \frac{1}{m_i} \sum_{j=1}^{m_i} \mathbf{e^i_j} \tag{6.1}$$

The vector representation for a bi-gram $\langle w^i\, w^{i+1} \rangle$ is acquired through:

$$\frac{1}{2}(\mathbf{e^i} + \mathbf{e^{i+1}}) \tag{6.2}$$

For instance, consider the sentence in Bengali: **হোটেলের খাওয়া দাওয়া দুর্দান্ত ভালো** (*hoteler khāowā dāowā durdānto bhālo*: The food is excellent at the hotel). This sentence is tokenized as **হোটেলের, খাওয়া, দাও, ##য়া, দুর্দান্ত, ভালো** (*hoteler, khāowā,*

dāo, ##wā, durdānto, bhālo). Notably, the word দাওয়া (*dāowā*) is split into two tokens, দাও(*dāo*) and ##য়া (*##wā*) by the Bengali BERT model. The individual embeddings of tokens from the BERT model are illustrated in Fig. 6.1.

Cosine Similarity: Cosine similarity is a metric used to determine how similar two vectors, **A** and **B**, are in a multidimensional space. This metric is particularly useful in fields like natural language processing, where it helps in comparing the similarity of textual data represented as vectors.

The cosine similarity between two vectors is calculated using the following formula:

$$\text{cosin}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{6.3}$$

In this equation, the numerator represents the dot product of vectors **A** and **B**. The dot product essentially measures the degree to which these two vectors align in the same direction. If the vectors are pointing in the exact same direction, the dot product will be positive and high, indicating strong similarity. Conversely, if the vectors point in opposite directions, the dot product will be negative, suggesting dissimilarity.

The denominator of the formula consists of the product of the magnitudes (or Euclidean norms) of the two vectors, **A** and **B**. The magnitude of a vector is calculated as the square root of the sum of the squares of its components. This part of the formula normalizes the dot product, ensuring that the similarity measure is scale-invariant. After normalization, the cosine similarity value falls within a range of -1 to 1:

(a) A cosine similarity of **1** indicates that the vectors are identical, meaning they point in the exact same direction.

(b) A cosine similarity of **0** means that the vectors are orthogonal, indicating no similarity or correlation.

(c) A cosine similarity of **-1** implies that the vectors are diametrically opposed, indicating they are completely dissimilar.

This measure is particularly effective for high-dimensional data, where traditional distance metrics like Euclidean distance may not perform well. Cosine similarity is often used in information retrieval, text mining, and other applications where the focus is on the orientation of data rather than its magnitude.

For example, the cosine similarity between the embedding of Bengali reduplication খাওয়া দাওয়া (*khāowā dāowā*: fooding) in the sentence লড়াই লেগেছে, বলে তো আর খাওয়া দাওয়া বন্ধ থাকবে না (*larāi legeche bole to ār khāwā dāwā bondho thākbe nā*:

When there's conflict, food service won't be discontinued) and the embedding of খাওয়া দাওয়া (*khāowā dāowā*) in the sentence হোটেলের খাওয়া দাওয়া দুর্দান্ত ভালো (*hoteler khāowā dāowā durdānto bhālo*: The food is excellent at the hotel) is 97.46%.

6.2 Creation of Sense Database

The process of mapping reduplications to their corresponding senses within a Bengali text corpus involves creating a predictive model through two distinct but interconnected phases.The steps for both the phases are outlined in the Flowchart depicted in Fig. 6.2.

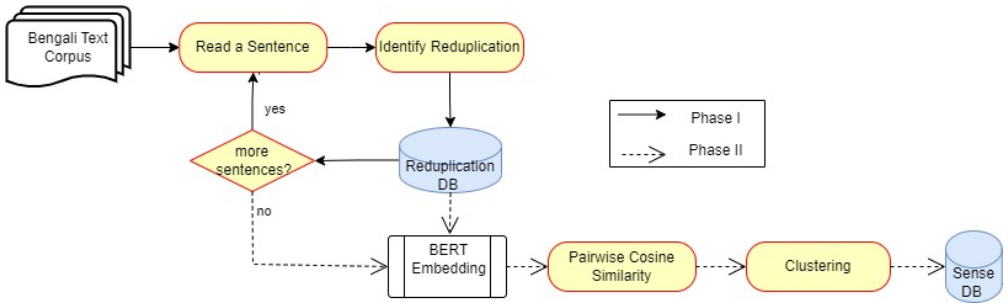

Figure 6.2: Creation of Sense Model

Phase I: Building the Reduplication Database

i. **Corpus Analysis:** The initial step involves examining each document in the corpus. Each sentence within these documents is analyzed to identify instances of reduplication.

ii. **Identification and Storage:** For every sentence in which reduplications are found, these instances are recorded. Each reduplication, along with its context (i.e., the sentence in which it appears), is stored in a database named *Reduplication DB*. This database serves as a comprehensive repository of all reduplications and their occurrences within the corpus.

Phase II: Sense Mapping and Clustering

i. **Generating Embeddings:** With *Reduplication DB* populated, the next phase involves creating BERT embeddings for each reduplication within the context of its sentence. BERT (Bidirectional Encoder Representations from Transformers) generates vector representations that capture the contextual meaning of each reduplication.

ii. **Computing Cosine Similarity:** Once embeddings are generated, the cosine similarity between each pair of these embeddings is calculated. Cosine similarity measures how similar two vectors are, with a value of 1 indicating perfect similarity, 0 indicating no similarity, and -1 indicating perfect dissimilarity. This measure helps in grouping similar reduplications into clusters.

iii. **Clustering:** The embeddings are clustered based on their cosine similarity. Each cluster represents a distinct sense or meaning of the reduplication. For instance, if two reduplications are used in similar contexts and have similar embeddings, they are likely to belong to the same cluster, indicating they share a similar sense.

Density-Based Clustering (DBSCAN) is chosen for this task because the number of clusters is unknown, and outliers are present. Unlike partition-based clustering methods such as K-Means, DBSCAN does not require specifying the number of clusters beforehand, making it well-suited for unsupervised classification of reduplications. Since partial reduplications typically carry a single sense and do not form high-level semantic categories, a density-based approach helps group similar instances while isolating noise and outliers effectively.

The MinPts parameter is set to twice the average number of sentences per reduplication in the corpus, ensuring that clusters form only when reduplications appear frequently enough across different contexts. The Epsilon (ϵ) value, which determines the neighborhood radius for clustering, is defined as twice the average distance between instances of a reduplication appearing in different sentences, helping to capture semantic consistency. To measure the similarity between reduplication instances, cosine similarity is used, with distance computed as (1 - cosine similarity) between two embeddings, ensuring that semantically close reduplications are grouped together while dissimilar ones remain apart.

iv. **Calculating Centroids and Thresholds:** Within each cluster, a centroid is determined by averaging the vector components of all reduplications in that cluster. The centroid

represents the central vector of the cluster, which is used to quantify the common sense shared by the reduplications in that cluster. For each vector, the cosine similarity to the cluster centroid is computed, and the average of these similarities is established as the threshold value (T_h). This threshold is crucial for real-time sense prediction.

v. **Storing the Sense Model:** Finally, the cluster centroids, their associated reduplications, and the threshold values are stored in a separate database named *Sense DB*. This database serves as a reference model for predicting the sense of reduplications in new texts.

By maintaining and utilizing these databases—*Reduplication DB* and *Sense DB*—the system can accurately tag reduplications in real-time, thereby enhancing the understanding of their meanings in various contexts.

6.3 Realtime Sense Tagging

The process focuses on the real-time identification of reduplications within sentences and their corresponding sense tagging. This procedure is essential for ensuring that reduplications are accurately interpreted in the context of the sentence they appear in, whether the text comes from newspapers, social media posts, or other sources. The detailed steps and processes involved in Real-time Sense Tagging are comprehensively illustrated in Fig. 6.3.

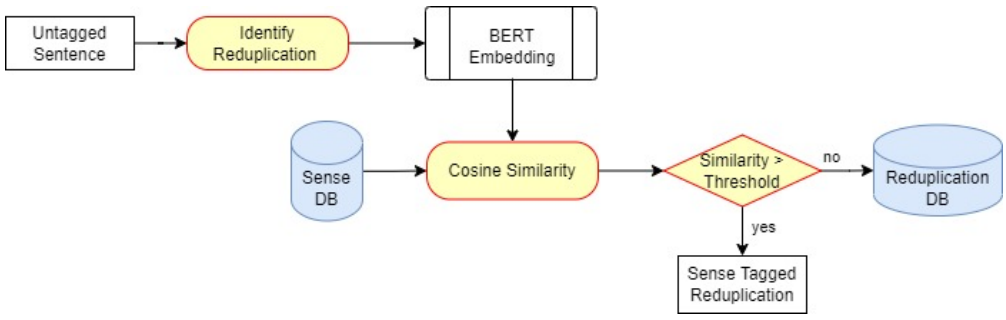

Figure 6.3: Sense Tagging and Evaluation of Sense Model

i. **Identification of Reduplications:** The process begins with receiving a sentence, which could be sourced from various platforms such as newspapers, social media posts, blogs, or other textual content.The system scans the sentence to identify any instances of reduplication. Reduplication refers to a linguistic phenomenon where a word or part of a word is repeated, often with slight modifications.

ii. **BERT Embedding Calculation:** For each identified reduplication, the system computes its BERT (Bidirectional Encoder Representations from Transformers) embedding. This embedding is a high dimensional numerical vector that encapsulates the contextual meaning of the reduplication within the specific sentence. Typically, each embedding is a high-dimensional vector (e.g., 768 dimensions) that represents the semantic content of the reduplication.

iii. **Cosine Similarity Computation:** The system calculates the cosine similarity between the embedding of the identified reduplication and each cluster centroid stored in the Sense Database (*Sense DB*). The cluster centroids represent different senses of reduplications, and the similarity measure helps in determining how close the current reduplication is to these predefined senses. Cosine similarity ranges from -1 to 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity.

iv. **Sense Tagging Based on Threshold:** The system identifies the cluster centroid with the highest cosine similarity score. If this maximum similarity exceeds a predefined threshold (T_h), the system considers the match to be strong enough to assign the corresponding sense from the Sense DB to the identified reduplication. The identified sense is then tagged to the reduplication within the sentence, making the sentence semantically richer and more informative.

v. **Handling of Low Similarity Cases:** If the highest similarity score does not surpass the threshold, the system flags the instance as an error, indicating that it cannot confidently assign a sense to the reduplication. The sentence and its identified reduplication are then stored in the Reduplication Database for future analysis. This helps in understanding why the tagging failed and what improvements can be made to the model.

vi. **Dynamic Nature of Reduplications:** Reduplications are constantly evolving in the language used by native speakers, leading to the creation of new reduplications and new senses. The errors identified in the process serve as potential indicators for these new linguistic developments and help in identifying new reduplications and their associated senses.

vii. **Periodic Update of the Sense Database:** To keep the Sense Database (*Sense DB*) and the prediction model up-to-date, a periodic recreation process is implemented. This involves analyzing newly identified reduplications and incorporating them into the *Sense DB*. The periodic updates ensure that the system remains accurate and reflective of the latest linguistic patterns,

automatically improving its performance in tagging reduplications over time. This process allows the system to adapt to new reduplications and their senses, maintaining the relevance and accuracy of the sense tagging model.

6.4 Identified Senses

In the study, an in-depth analysis was conducted to understand how reduplicated words are represented in vector space when embedded using word embeddings, particularly BERT embeddings. The focus was on identifying the specific meanings, or "senses," that these reduplicated words convey within different sentence contexts.

When words that involve partial or complete reduplication are embedded into vector space from various sentences, they tend to cluster together naturally. This clustering implies that a reduplicated expression carries a consistent meaning or sense across all the sentences in which it appears. For example, the phrase খাওয়া দাওয়া (*khāowa dāowa*: fooding) consistently clusters in vector space, regardless of the sentences in which it appears. This indicates that the reduplication structure plays a significant role in conveying a specific, recognizable meaning.

For words that are completely reduplicated, the clustering often pulls in not just identical forms but also related reduplicated forms. This suggests that full reduplication doesn't just duplicate meaning but often extends it, encompassing a wider array of interpretations or senses.

To provide a clearer understanding, the study categorizes the various senses that reduplication can convey, as outlined below:

(a) *Onomatopoeia or Sound*: Reduplication can represent sounds or onomatopoeic words, such as ঝন ঝন (*jhan jhan*: jingling), খল খল (*khal khal*: sound of laughter), and ধূপ ধূপ (*dhup dhup*: sound of falling).

(b) *Continuous and Incomplete Action*: Reduplication can indicate actions that are continuous or incomplete, as seen in words like চিবোতে চিবোতে (*chibote chibote*: chewing), কাপতে কাপতে (*kapte kapte*: trembling), and লাফাতে লাফাতে (*lafate lafate*: jumping).

(c) *Irregular Occurrences*: Reduplication may denote irregular occurrences or sudden actions, for instance, মাঝে মাঝে (*majhe majhe*: occasionally) and হঠাৎ হঠাৎ (*hotath hotath*: abruptly).

(d) *Sense of Plurality*: Reduplication can convey a sense of plurality, such as বড় বড় (*bara bara*: many big things) and লম্বা লম্বা (*lamba lamba*: many long things).

(e) *Sense of Intensity*: It can also indicate an increased degree of intensity, as seen in গরম গরম (*garom garom*: hotter than usual) and ঠান্ডা ঠান্ডা (*thanda thanda*: colder than usual).

(f) *Sense of Repetition*: Reduplication can emphasize repetition, such as বছর বছর (*bachar bachar*: every year) and প্রহরে প্রহরে (*prahare prahare*: every hour).

(g) *Emotion or Sentiment*: It can express emotions or sentiments, as in হাসি হাসি (*hasi hasi*: expressing happiness) and কান্না কান্না (*kanna kanna*: expressing sorrow).

The findings underscore the significant role of reduplication in conveying appropriate meanings in language. By clustering together in vector space, these reduplications reveal distinct senses that are integral to understanding the broader context of the text. This detailed categorization of senses not only enhances the understanding of reduplication but also provides a structured approach to tagging and interpreting these linguistic features in real-time applications.

6.5 Results and Analysis

This experiment makes use of two Bengali corpora: the "Bengali Corpus from TDIL" and the "Bengali Corpus from Wikipedia," as described in Section 3.1.

Model Development and Testing Approach: For each of the corpora, a 4-fold cross-validation technique is employed to evaluate the robustness and generalization ability of the model. The entire dataset contained in *Reduplication DB* is evenly divided into four subsets, with each subset representing 25% of the data. This approach ensures that every data point is used for both training and testing over the course of the evaluation process. During each iteration of cross-validation:

(a) One of the four subsets (25% of the data) is designated as the test set for validation.

(b) The remaining three subsets (75% of the data) are combined to form the training set, which is used to build the model or construct the Sense Database.

This process is repeated four times, with each subset serving as the test set exactly once. By doing so, the model is trained and validated on different portions of the data, ensuring that the evaluation is not biased towards any particular subset.

Table 6.1: Performance Comparison for Sense Tagging

Dataset	BERT Model	Precision (%)	Recall (%)	F1-Score (%)
TDIL	Bengali BERT	90.90	66.67	76.92
TDIL	Multilingual BERT	89.26	55.56	68.49
Wiki	Bengali BERT	93.33	77.78	84.96
Wiki	Multilingual BERT	86.79	51.11	64.33

The average performance across all four folds provides a reliable estimate of the model's effectiveness. The best performing model among the four folds is finally saved for real-time sense tagging.

Embedding Strategy and Performance Evaluation: The accuracy of the sense tagging system was evaluated by varying the embedding strategy and utilizing two different BERT models:

(a) *Bengali BERT*: A BERT model pretrained specifically on Bengali text, tailored to the linguistic nuances of the Bengali language.

(b) *Multilingual BERT*: A BERT model pretrained on multiple languages, including Bengali, but not exclusively focused on it.

The impact of using a language-specific model versus a general multilingual model was assessed by independently testing these models on each dataset. The performance metrics from these experiments are summarized in Table 6.1.

Insights from the Results: The results presented in Table 6.1 provide clear evidence that the system's performance is significantly influenced by the choice of the pretrained BERT model:

(a) *Bengali BERT* consistently outperformed *Multilingual BERT* across both datasets, achieving higher F1-score.

(b) For the *TDIL corpus*, Bengali BERT achieved a F1-score of 76.92% compared to 68.49% for Multilingual BERT.

(c) For the *Bengali Wikipedia* dataset, the performance gap was even more pronounced, with Bengali BERT achieving F1-score of 84.96% , while Multilingual BERT lagged behind with scores of 64.33%.

These findings underscore the importance of using language-specific models like Bengali BERT to achieve superior results in natural language processing tasks. The superior performance of Bengali BERT highlights its ability to capture the

unique linguistic characteristics of Bengali, making it more effective for tasks like sense tagging in this language.

The experimental results conclusively demonstrate that the performance of the sense tagging system is greatly enhanced when utilizing a BERT model that is specifically pretrained on the Bengali language. This language-specific approach not only improves accuracy but also ensures that the system is better equipped to handle the intricacies and nuances of Bengali text, leading to more reliable and meaningful sense tagging outcomes.

6.6 Conclusion

This chapter details the development of an advanced algorithm for attributing specific senses to reduplications in Bengali texts, leveraging the Bengali BERT model for enhanced accuracy in NLP tasks. Emphasizing the semantic nuances of reduplication in Bengali, the study introduces a dynamic, real-time mechanism that updates the reference model with new data to ensure adaptability and sustained accuracy in evolving language contexts.

CHAPTER 7

Conclusion and Future Work

This chapter concludes by addressing the range of challenges faced throughout different phases of the experimental process, while also highlighting potential directions for future research. Some of the obstacles encountered had a relatively minor impact on the system's performance, and with adequate effort, these could be effectively mitigated. However, a significant number of challenges had a profound effect on performance, and these could not be entirely resolved, as they stem from the intrinsic nature of the language itself. To achieve more substantial improvements in system performance, future research should focus on isolating and addressing the specific issues individually. By conducting targeted experiments on these distinct problems, it may be possible to develop more effective solutions that can better handle the intricate challenges presented by the language, ultimately leading to significant advancements in system performance.

7.1 Conclusion of the Present Works

In this study, various rule-based and machine learning based approaches for identification of reduplication has been explored. A novel association measure is explored that utilizes Levenshtein Distance combined with *Word Expansion* to identify reduplications. It is observed that carefully crafted rules yeilds an impressive performance on identification of reduplication in Bengali.

This rules are combined with other statistical and linguistic features using various Machine Learning algorithms the system perfomnace improves for identification of reduplications in Bengali.

To explore the generative nature of reduplicated form computationally, Finite State Transducers are employed. Finite State Transducers are the mathematical

machine model for generation of reduplications. This FSTs are also imployed to identify reduplications and that too gives impressive performance.

Various senses reduplications convey are also explored in this research. Senses of reduplications depends on the contex they are appearing. To analyse the context BERT models are used. The proposed system builds a repository for various senses of reduplications in Bengali. Since, reduplications are generative in nature, a provision has been made to the system such that the sense repository may grow incrementally over time to make the system robust.

7.2 Challanges Faced

Processing Multi-Word Expressions (MWEs) presents several challenges in the field of natural language processing. Here are some of the key challenges:

i. **Generative Nature:** Bengali reduplications are not static; they evolve with the language, reflecting changes in culture, society, and communication. For instance, the reduplication খাওয়া দাওয়া (*khāoā dāoā*: food and drink) has long been used in Bengali. However, as new cultural phenomena emerge, new reduplications may arise to capture these changes. For example, in the context of modern technology, a phrase like পোস্ট টোস্ট (*post tost*: posting things on social media) could emerge, reflecting new digital behaviors. As the language evolves, it requires continuous updates to the models and algorithms that identify and interpret it. Keeping pace with the dynamic nature of language use is an ongoing challenge, as it requires regularly refreshing linguistic databases and adapting to newly emerging patterns in real-time language processing.

ii. **Proper Nouns:** Proper nouns can sometimes exhibit structural features that mimic the appearance of reduplications, leading to potential challenges in accurate linguistic classification. For instance, names such as রাম রায় (Ram Roy) and কোকা কোলা (Coca Cola) bear a superficial resemblance to reduplicative patterns. Classifying such instances as reduplications can introduce false positives, thereby compromising the accuracy and reliability of the proposed approach.

iii. **Reduplications with Function Words:** In Bengali, reduplication often involves not just content words (such as nouns, verbs, or adjectives) but also function words (such as prepositions, conjunctions, and pronouns). For instance, reduplication of function words can be seen in examples like সঙ্গে সঙ্গে (*songe songe*: immediately). In this case, a function word is repeated to emphasize immediacy or urgency, thereby creating a nuanced meaning.

However, when performing natural language processing (NLP) tasks, such as text analysis or feature extraction, it is common practice to remove stop words (i.e., function words) from the data to reduce the dimensionality of feature vectors. This reduction in dimensionality is crucial for managing computational complexity and improving the performance of machine learning models. Unfortunately, this approach also removes reduplicated function words, which can lead to the loss of important linguistic patterns or meanings that rely on these repetitions. Here, these function words act more like content words, contributing to the meaning of the sentence rather than just serving a grammatical function. This duality creates a challenge: while removing function words can simplify data processing, it can also strip the text of meaningful patterns, especially in languages like Bengali where reduplication plays a significant role.

iv. **Orthographic Variability:** One of the significant challenges in processing reduplicated expressions in Bengali lies in their inconsistent orthographic representations across various textual sources. Reduplicated words may appear with spaces (e.g., সামনা সামনি (*sāmnā sāmni*)), with hyphens (e.g., সামনা-সামনি (*sāmnā-sāmni*)), or without any delimiters (e.g., সামনাসামনি (*sāmnāsāmni*)). These surface-level variations can interfere with tokenization, rule-based identification, and dictionary lookups—especially in systems that rely on strict string matching or fixed patterns. Consequently, accurate recognition of reduplication becomes more challenging, potentially reducing the performance of information retrieval and natural language processing tasks. Addressing these orthographic inconsistencies is crucial to improving the robustness and effectiveness of reduplication detection in Bengali.

v. **Stemming and Lemmatization:** In morphologically rich languages like Bengali, stemming and lemmatization are essential for extracting root words. However, they can interfere with the structure and meaning of reduplicated expressions. For instance, the partial reduplication খবরা খবর (*khoborā khobor*) may be reduced to খবর খবর (*khobor khobor*), incorrectly representing it as a full reduplication. This transformation distorts the original meaning and can introduce structural inaccuracies in downstream NLP tasks. As a result, such disruptions diminish the performance of systems that depend on syntactic or semantic precision, underscoring the need for reduplication-aware morphological processing in Bengali NLP.

vi. **Spelling Mistakes:** Spelling errors present a significant challenge in automated systems, as they can lead to misinterpretation or omission of important data. Unlike humans, who can easily recognize and correct such errors, auto-

mated systems rely on exact word matches, causing a drop in accuracy and performance when typographical mistakes occur.

vii. **Long Sentences:** Long sentences in Bengali pose challenges for natural language processing as they often include irrelevant or excessive information, making it hard for models like BERT to focus on the core meaning. Consider the following long sentence:
গতকাল সন্ধ্যায় যখন আমি কাজ থেকে বাসায় ফিরছিলাম, তখন রাস্তায় হালকা বৃষ্টি পড়ছিল, আর আমি ধীরে ধীরে হাঁটতে হাঁটতে আশপাশের মানুষদের ব্যস্ততায় মগ্ন দেখছিলাম, কারণ সবাই যেন দ্রুত তাদের গন্তব্যে পৌঁছানোর চেষ্টা করছিল, কিন্তু আমি বরং শহরের এই ভেজা রাতের সৌন্দর্য উপভোগ করতে চেয়েছিলাম (*gatakāl shandhāy jakhan ami kaj theke bashay firchilam, takhan rāstāy hālkā brishti parchhila, ar ami dhire dhire hānte hānte āshpāsh-er mānushder byastatāy magna dekhchilam, kāron shabāi jena druta tāder gantabbe pauchhānor cheshtā karchhila, kintu āmi barang shaharer ei bheja rāter shaundarya upabhog karte cheyechilām*: Last evening, when I was returning home from work, there was a light drizzle on the street, and as I slowly walked along, I watched the people around me absorbed in their busy lives, all seemingly trying to reach their destinations quickly, but I, on the other hand, wanted to enjoy the beauty of the city's wet night).

viii. **Short Sentences:** Handling short sentences in Bengali reduplication poses challenges due to their limited contextual information, which affects models like BERT embeddings that rely on context for accurate interpretation. For instance, in the sentence সে চুপচাপ। (*se chupchap*: He/She is extremely quiet.), the lack of surrounding context can cause the model to miss the emphasis added by the reduplication, leading to an incomplete or incorrect understanding of the sentence.

7.3 Close Observation of Thesis Results and Possible Improvements

This thesis presents an in-depth study of Bengali reduplication, focusing on three key aspects: Identification, Generation, and Sense Tagging. The results achieved in this research are highly promising; however, there is still room for further improvement. A detailed analysis of the obtained results has been conducted, identifying potential areas for enhancement, which are also discussed in this study.

Reduplication Identification: The rule-based method that uses *Word Expansion* and Levenshtein distance for identifying reduplications achieves an F1-score

of 87.80%, which is quite impressive considering that a single rule cannot fully capture the wide variety of reduplications found in Bengali.

The proposed approach, designed to identify partial reduplications in Bengali text, exhibits certain limitations when two specific conditions co-occur within the same bi-gram. These conditions arise when the first letter of one word in the bi-gram is a vowel, and its corresponding allograph attaches to the initial consonant of the other word. Additionally, mismatches between the words can occur in later segments of the bi-gram. For example, consider the bi-gram উসখো খুসকো (*ushkhō kushkō*: disheveled). In this case, the vowel উ (*u*) functions as an allograph of the consonant খ (*kha*), and a mismatch appears at the third character within the bi-gram. After applying the *Word Expansion* technique, the bi-gram transforms into উসখও খউসকও. Despite this transformation, the Levenshtein distance between the two words remains 2, which prevents the proposed rule based method from correctly detecting উসখো খুসকো (*ushkhō kushkō*) as a partial reduplication. A similar limitation occurs with the bi-gram পরীক্ষা নিরীক্ষা (*parīkṣā nirīkṣā* : experiment). In this instance, two key conditions are observed: the initial consonant shifts from প (*pa*) to ন (*na*), and the vowel ঈ (*ī*) is added. The bi-grams উসখো খুসকো (ushkhō kushkō) and পরীক্ষা নিরীক্ষা (*parīkṣā nirīkṣā*) result in False Negative (FN) errors for the rule-based method.

Additionally, some proper nouns exhibit structural similarity to reduplications, even though they are not true reduplications, and are incorrectly classified as reduplications. Examples include রাম রায় (Ram Roy) and কোকা কোলা (Coca Cola). The bi-grams রাম রায় (Ram Roy) and কোকা কোলা (Coca Cola result in False Positive (FP) errors for the rule-based method.

However, due to this limitation, a machine learning-based approach is explored, which is trained using 32 distinct features, that performed better with an F1-score of 93.02%. The bi-grams রাম রায় (Ram Roy) and কোকা কোলা (Coca Cola) also contribute to False Positive (FP) errors in the machine learning-based method.

However, there is an opportunity to enhance the model's performance by incorporating additional linguistic rules. These rules can be integrated as features in the machine learning framework, allowing the model to learn from both data-driven patterns and rule-based insights. By combining these approaches, the identification of reduplications can be further refined, improving overall accuracy and adaptability to diverse linguistic structures. Additionally, the use of POS-tagged corpora would provide valuable syntactic context to the model, helping it distinguish between true reduplications and structurally similar word pairs that do not serve a reduplicative function. For example, certain noun-adjective combinations or proper noun pairs might share surface-level similarities with reduplications but function very differently within sentences. By analyzing the Part-of-Speech (POS)

sequences, the model could learn to differentiate between content-based reduplications and incidental co-occurrences.

Reduplication Generation: A total of 20 linguistic patterns [90] have been identified and utilized to develop a computational model for Bengali reduplication generation. These patterns serve as fundamental rules that guide the model in generating reduplicated words accurately. Since reduplication in Bengali exhibits diverse structural variations, these predefined patterns help in capturing common forms effectively.

To assess the effectiveness of this generation process, the model was tested on its ability to correctly identify partial reduplications, achieving an F1-score of 87.81%. This indicates a strong performance; however, the model still has room for improvement. One of the key challenges is that Bengali reduplication follows complex and diverse patterns, some of which may not be fully captured by the existing 20 rules. For instance, none of the 20 predefined rules cover the generation of the reduplication কাপড় চোপড় (*kāpaṛ copaṛ*: clothing).

To further enhance the performance of the model, additional linguistic research is needed to identify new reduplication patterns in Bengali. Incorporating more patterns into the computational framework could significantly improve the model's accuracy and robustness.

Sense Tagging of Reduplication: The sense tagging of reduplication in Bengali achieves an F1-score of 84.96%, indicating a promising but improvable performance. The accuracy of this system depends on two critical factors: the effectiveness of reduplication identification and the ability of the underlying BERT model to capture contextual meaning accurately. For example, Bengali BERT and Multilingual BERT exhibit different levels of performance when applied to the same Bengali text corpus.

One possible way to improve the accuracy is by incorporating larger and more advanced models, such as LLM-based architectures (e.g., GPT or fine-tuned transformer models with a larger parameter space). These models have a deeper contextual awareness, which could lead to more precise sense tagging. Additionally, the accuracy of sense tagging is directly dependent on reduplication identification accuracy. If the reduplication identification step is enhanced, ensuring fewer errors in detecting reduplicated forms, the overall performance of sense tagging will also improve significantly. Future work could explore data augmentation techniques, fine-tuned transformer models, and hybrid approaches to further optimize the sense tagging of reduplication in Bengali NLP.

7.4 Scope for Future Works

Future research holds considerable potential for enhancing the accuracy of reduplication identification and analysis in Bengali by developing and implementing new rules and computational techniques. These future advancements can leverage innovative association measures, enriched linguistic features, and advanced computational approaches, combining linguistic insights with cutting-edge machine learning (ML) and deep learning methods.

I. **Expanding Linguistic Rule Coverage**
The current study relies on a predefined set of linguistic rules for detecting reduplication, but Bengali reduplication encompasses numerous patterns not yet covered by these rules. Future research could focus on systematically analyzing diverse textual sources, including literature, colloquial speech, and social media content, to identify new reduplication patterns. Incorporating these patterns into both rule-based systems and hybrid machine learning models would improve coverage and detection accuracy.

II. **Enhancing Machine Learning Algorithm Performance**
The current machine learning models for reduplication detection can be further improved by:

- Incorporating newly identified linguistic rules as additional features.
- Expanding feature sets to include morpho-syntactic patterns, phonetic similarity measures, and POS-tag sequences.
- Developing ensemble models that combine predictions from rule-based systems and different machine learning algorithms to improve robustness and accuracy.
- Leveraging semi-supervised learning techniques to exploit large unlabeled Bengali corpora, especially in low-resource contexts.

III. **Exploring Other Multi-Word Expressions (MWEs)**
This research focuses specifically on reduplication, but future research can explore other types of MWEs prevalent in Bengali. These include:

- *Idiomatic Expressions*, such as আকাশ থেকে পড়া (*ākāsh theke paṛā*: to be very surprised), which have non-compositional meanings.
- *Proverbs*, such as যত গর্জে তত বর্ষে না (*jata garje tata barṣe na*: A barking dog seldom bites), which encode cultural knowledge.

Table 7.1: Relevant LLMs for Bengali Reduplication

Model	**Organization**	**Strengths**
GPT-4 Turbo	OpenAI	Multilingual, strong in context understanding
mBERT	Google	Multilingual, trained on Bengali text
XLM-RoBERTa	Facebook AI	Strong multilingual performance
BengaliBERT	Hugging Face	Specifically trained on Bengali
Gemini	Google	Multimodal and multilingual
Mistral	Mistral AI	Efficient and multilingual

- *Light Verb Constructions (LVCs)*, such as সিদ্ধান্ত নেওয়া (*siddhānta newā*: to make a decision), where the verb carries limited meaning.
- *Phrasal Verbs*, such as ফিরে আসা (*fire āsa*: to return), where the combination of verb and particle alters the meaning.

Studying these MWEs will contribute to a more comprehensive understanding of complex expressions in Bengali, benefiting machine translation, information retrieval, and semantic analysis.

IV. **Leveraging Large Language Models (LLMs):** The use of Large Language Models (LLMs), including GPT-based and multilingual transformer models, presents promising opportunities for the identification and sense tagging of Bengali reduplication.

- **Reduplication Identification:** Prompt engineering can be leveraged to guide LLMs toward accurate identification of Bengali reduplications. This can be done by designing structured prompts that clearly specify the task and provide relevant context. Additionally, incorporating few-shot examples where a small set of labeled examples are included within the prompt itself helps the model better understand the expected output format and semantic distinctions. Some of the LLMs models useful for Bengali Reduplication analysis are specified in Table 7.1.
- **Sense Tagging of Reduplication:** Large Language Model (LLM)-generated embeddings, with their high-dimensional vectors, play a critical role in capturing not just the meaning of individual words, but also the broader contextual relationships within a sentence or document. This capability is particularly important for complex linguistic phenomena such as sense tagging and semantic analysis in Bengali

reduplication, where the meaning of reduplicated forms often depends heavily on the surrounding words and overall context.

Modern language models, like LLaMA 3, showcase the continuous advancement of embedding techniques. With embedding dimensions extending up to 4096, these models effectively process longer contexts, enabling them to capture meaning across extended sentences, paragraphs, or entire documents. In contrast, this research utilizes Bengali BERT and Multilingual BERT, both of which employ embedding vectors of 768 dimensions.

As embedding dimensions continue to grow in cutting-edge models, the ability to handle rich linguistic phenomena like reduplication will improve, opening new research opportunities in low-resource language processing.

Overall, future studies that combine linguistic insights, advanced computational techniques, enriched datasets, and the power of large language models could significantly advance the understanding, detection, generation, and interpretation of Bengali reduplications and other complex MWEs, contributing to the broader field of computational linguistics for low-resource languages.

Bibliography

[1] C. Fellbaum, "Towards a representation of idioms in WordNet," in *Usage of WordNet in Natural Language Processing Systems*, 1998.

[2] B. B. Chaudhuri, *Bangla Dhvanipratik: Svarup o Abhidhan (Bengali Sound Symbolism: Properties and Dictionary)*. Kolkata: Paschimbanga Bangal Akademi, 2010.

[3] S. Inkelas, "The dual theory of reduplication," *Linguistics*, vol. 46, no. 2, pp. 351–401, 2008.

[4] N. Dash, *Chapter 8 Structure of Reduplicated Forms in Bengali*. Oct. 2023, pp. 225–254, ISBN: 978-1-107-06424-9.

[5] A. Barman, D. Saha, and A. R. Pal, "An approach for maintaining structural uniformity of multiword expressions," in *2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Bhilai, India, 2024, pp. 1–4. DOI: 10.1109/ICAECT60202.2024.10469553.

[6] A. Barman and D. Saha, "Algorithm for removal of semantically insignificant content words," *International Journal of Computer Sciences and Engineering*, vol. 07, no. 01, pp. 53–56, 2019.

[7] A. Barman, D. Saha, and A. R. Pal, "An upgraded approach for identifying partially reduplicated forms in bengali text," *SN Computer Science*, vol. 05, no. 07, p. 892, 2024. DOI: 10.1007/s42979-024-03069-9.

[8] A. Barman, D. Saha, and A. R. Pal, "Bengali reduplication generation with finite-state transducers (fsts)," *International Journal of Speech Technology*, vol. 27, no. 03, pp. 729–737, 2024. DOI: 10.1007/s10772-024-10124-6.

[9] J. K. Raulji and J. R. Saini, "Stop-word removal algorithm and its implementation for sanskrit language," *International Journal of Computer Applications*, vol. 150, no. 2, Sep. 2016, ISSN: 0975-8887.

[10] V. Jha, N. Manjunath, P. D. Shenoy, and K. R. Venugopal, "Hsra: Hindi stopword removal algorithm," in *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*, Durgapur, India, 2016, pp. 1–5.

[11] S. Siddiqi and A. Sharan, "Construction of a generic stopwords list for hindi language without corpus statistics," *International Journal of Advanced Computer Research*, vol. 8, no. 34, pp. 35–40, 2017.

[12] R. R. M. and S. J. R., "A rule-based approach to identify stop words for gujarati language," in *Advances in Intelligent and Soft Computing (AISC) Series*, Springer-Verlag, 2017.

[13] M. P. Sinka and D. W. Corne, "Evolving better stoplists for document clustering and web intelligence," in *Design and Application of Hybrid Intelligent Systems*, 2003, pp. 1015–1023.

[14] R. Al-Shalabi, G. Kanaan, J. M. Jaam, A. Hasnah, and E. Hilat, "Stopword removal algorithm for arabic language," in *Proceedings of the 2004 International Conference on Information and Communication Technologies: From Theory to Applications*, Damascus, Syria, 2004, p. 545.

[15] B. Alhadidi and M. Alwedyan, "Hybrid stop-word removal technique for arabic language," *Egyptian Computer Science Journal*, vol. 30, no. 1, pp. 35–38, 2008.

[16] L. Dolamic and J. Savoy, "When stopword lists make the difference," *Journal of the American Society for Information Science and Technology*, no. 1, pp. 200–203, 2009.

[17] R. Puri, R. P. S. Bedi, and V. Goyal, "Automated stopwords identification in punjabi documents," *An International Journal of Engineering Sciences*, vol. 8, no. June 2013, pp. 119–125, 2013.

[18] A. T, K. M, and P. P, "Pre-processing phase of text summarization based on gujarati language," *International Journal of Innovative Research in Computer Science Technology (IJIRCST)*, vol. 2, no. 4, Jul. 2014.

[19] D. Sharma and S. Jain, "Evaluation of stemming and stop word techniques on text classification problem," *International Journal of Scientific Research in Computer Science and Engineering*, vol. 3, no. 2, pp. 1–4, Apr. 2015.

[20] S. Pan and D. Saha, "An automatic identification of function words in tdil tagged bengali corpus," *International Journal of Computer Sciences and Engineering*, vol. 07, no. 01, pp. 20–27, 2019.

[21] D. Osman and H. Bilge, "Effects of dimensionality reduction and feature selection in text classification," in *2011 IEEE 19th Signal Processing and Communications Applications Conference (SIU 2011)*, 2011, pp. 21–24. DOI: 10.1109/SIU.2011.5929577.

[22] A. Chakrabarty, A. Chaturvedi, and U. Garain, "A neural lemmatizer for bengali," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia: European Language Resources Association (ELRA), 2016, pp. 2558–2561.

[23] A. Chakrabarty and U. Garain, "Benlem (a bengali lemmatizer) and its role in wsd," in *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 15, Feb. 2016, Article 12, 18 pages. DOI: http://dx.doi.org/10.1145/28354941.

[24] K. Lindén and T. Pirinen, "Hfst tools for morphology - an efficient open-source package for construction of morphological analyzers," in *Proceedings of the Free/Open-Source Language Resources Workshop*, 2009, pp. 28–34.

[25] K. Darwish, "Building a shallow arabic morphological analyzer in one day," in *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, 2003, pp. 47–54.

[26] S. Sharoff, "Creating general-purpose corpora using automated search engine queries," in *WaCky! Working Papers on the Web as Corpus*, 2008, pp. 63–98.

[27] A. G. Jivani, "A comparative study of stemming algorithms," *International Journal of Computer Technology and Applications*, vol. 2, pp. 1930–1938, 2011.

[28] W. Kraaij and R. Pohlmann, "Viewing stemming as recall enhancement," in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996, pp. 40–48.

[29] J. Savoy, "A stemming procedure and stopword list for general french corpora," *Journal of the American Society for Information Science*, vol. 50, no. 10, pp. 944–952, 1999.

[30] S. Das, R. Pandit, and S. K. Naskar, "A rule based lightweight bengali stemmer," in *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, 2020.

[31] N. S. Dash, P. Bhattacharyya, and J. D. Pawar, "The wordnet in indian languages," in *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*, Springer, 2017.

[32] J. D. et al., "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019, pp. 4171–4186.

[33] T. B. et al., "Language models are few-shot learners," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 187–210.

[34] E. Brill, "Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging," in *Computational Linguistics*, 1995, pp. 543–565.

[35] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996, pp. 133–142.

[36] A. M. J. Lafferty and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2001, pp. 282–289.

[37] W. X. Z. Huang and K. Yu, "Bidirectional lstm-crf models for sequence tagging," in *arXiv preprint arXiv:1508.01991*, 2015.

[38] M. Peters, M. Neumann, M. Iyyer, *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long Papers)*, vol. 1, 2018, pp. 2227–2237.

[39] B. S. M. Marcus and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[40] N. Xue and L. Shen, "Chinese word segmentation as lmr tagging," in *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, 2003, pp. 176–179.

[41] N. Habash and O. Rambow, "Arabic tokenization, morphological analysis, and part-of-speech tagging in one fell swoop," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, 2005, pp. 573–580.

[42] M. C. S. Dandapat and A. Sarkar, "Part-of-speech tagging and chunking with crfs and transformation-based learning," in *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP)*, 2010, pp. 1–8.

[43] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999, ISBN: 978-0-262-13360-9.

[44] G. Nunberg, I. A. Sag, and T. Wasow, "Idioms," *Language*, vol. 70, pp. 491–538, 1994.

[45] B. Das, "Extracting collocations from bengali text corpus," *Procedia Technology*, vol. 4, pp. 325–329, 2012. DOI: 10.1016/j.protcy.2012.05.049.

[46] S. D. et al., "Statistical investigation of bengali noun-verb (nv) collocations as multi-word expressions," in *Proceedings of Modeling and Shallow Parsing of Indian Languages (MSPIL)*, 2006, pp. 230–233.

[47] D. Kiela and S. Clark, "Detecting compositionality of multi-word expressions using nearest neighbours in vector space models," in *EMNLP 2013 - Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA: Association for Computational Linguistics (ACL), 2013, pp. 1427–1432.

[48] S. Martens and V. Vandeghinste, "An efficient, generic approach to extracting multi-word expressions from dependency trees," in *Coling 2010 Organizing Committee*, Beijing, China, 2010, pp. 85–88.

[49] N. Dash, "Compound nouns and adjectives in bangla: Some empirical observations," in *MultiWord Workshop (MWW) at the AUKBC, Anna University*, Chennai, India, 2011.

[50] S. Mukhopadhayay, T. Dasgupta, M. Sinha, and A. Basu, "Automatic extraction of compound verbs from bangla corpora," in *Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing*, Mumbai, India: The COLING 2012 Organizing Committee, 2012, pp. 153–162.

[51] S. Paul, "Composition of compound verbs in bangla," in *Proceedings of the workshop on Multi-Verb constructions, Trondheim Summer School 2003*, Norwegian University of Science and Technology, Trondheim, 2003.

[52] M. S. Islam and J. K. Das, "Design analysis rules to identify proper noun from bengali sentence for universal networking language," *IJMECS*, vol. 6, no. 8, pp. 1–9, 2014.

[53] S. K. Choudhury and B. Kundu, "Convex: Conjunct verb extraction from parallel corpus: A hybrid approach," in *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, Kharagpur, 2012, pp. 1–6. DOI: 10.1109/IHCI.2012.6481852.

[54] K. Dutta and A. Jindal, "System for identification and analysis of reduplication words in hindi corpus," *International Journal of New Technology and Research*, vol. 2, no. 4, pp. 18–21, Apr. 2016.

[55] K. Nongmeikapam and S. Bandyopadhyay, "Identification of reduplication mwes in manipuri, a rule-based approach," in *Proceedings of the 23rd International Conference on the Computer Processing of Oriental Languages*, California, USA, 2010, pp. 49–54.

[56] P. Garg, A. Marwaha, and M. Goel, "Identification and classification of reduplication words in punjabi language," *International Journal of Scientific Technology Research*, vol. 9, no. 6, Jun. 2020.

[57] V. Gayen and K. Sarkar, "Automatic identification of bengali noun-noun compounds using random forest," in *Proceedings of the 9th Workshop on Multiword Expressions*, Atlanta, Georgia, USA: Association for Computational Linguistics, 2013, pp. 64–72.

[58] T. Chakraborty, "Using semantic clustering for detecting bengali multiword expressions," *International Journal of Linguistics and Language Resources (Lingvisticæ Investigationes)*, 2014, ISSN: 0378-4169.

[59] P. Pecina, "Reference data for czech collocation extraction," in *Proceedings of the LREC Workshop Towards a Shared Task for MWEs*, Marrakech, Morocco, 2008, pp. 11–14.

[60] W. Gharbieh, V. Bhavsar, and P. Cook, "Deep learning models for multiword expression identification," in *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, 2017. DOI: 10.18653/v1/S17-1006.

[61] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264.

[62] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 260–270. DOI: 10.18653/v1/N16-1030.

[63] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[64] M. S. I. Khondoker Ittehadul Islam and M. R. Amin, "Sentiment analysis in bengali via transfer learning using multi-lingual bert," in *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, 2020, pp. 1–5.

[65] A. Bhowmick and A. Jana, "Sentiment analysis for bengali using transformer based models," in *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, National Institute of Technology Silchar, Silchar, India: NLP Association of India (NLPAI), 2021, pp. 481–486.

[66] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[67] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[68] D. Pathak, S. Nandi, and P. Sarmah, "Reduplication in assamese: Identification and modeling," *ACM Transactions on Asian Language Information Processing*, vol. 21, pp. 1–18, 2022. DOI: 10.1145/3510419.

[69] K. R. Beesley and L. Karttunen, *Finite-state Morphology: Xerox Tools and Techniques*. Stanford: CSLI, 2003.

[70] H. Dolatian and J. Heinz, "Modeling reduplication with 2-way finite-state transducers," in *Proceedings of the 15th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2018, pp. 66–77.

[71] E. Filiot and P.-A. Reynier, "Transducers, logic and algebra for functions of finite words," *ACM SIGLOG News*, vol. 3, no. 3, pp. 4–19, 2016.

[72] V.-T. Bui and A. Savary, "Cross-type french multiword expression identification with pre-trained masked language models," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, Torino, Italia: ELRA and ICCL, 2024, pp. 4198–4204.

[73] A. Walsh, T. Lynn, and J. Foster, "A bert's eye view: Identification of irish multiword expressions using pre-trained language models," in *Proceedings of the 18th Workshop on Multiword Expressions @LREC2022*, Marseille, France: European Language Resources Association, 2022, pp. 89–99.

[74] A. Saif, N. Omar, U. Zainodin, and M. Aziz, "Building sense tagged corpus using wikipedia for supervised word sense disambiguation," *Procedia Computer Science*, vol. 123, pp. 403–412, 2018. DOI: 10.1016/j.procs.2018.01.062.

[75] A. Pal, D. Saha, N. Dash, S. Naskar, and A. Pal, "A novel approach to word sense disambiguation in bengali language using supervised methodology," *Sadhana*, vol. 44, pp. 1–12, 2019. DOI: 10.1007/s12046-019-1165-2.

[76] K. J. Devlin and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," vol. 1, pp. 4171–4186, 2019.

[77] S. Sarker, *Banglabert: Bengali mask language model for bengali language understanding*, 2020.

[78] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of Workshop at ICLR*, 2013.

[79] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is All You Need*. arXiv preprint arXiv:1706.03762, 2017.

[80] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," *EMNLP*, pp. 1532–1543, 2014. DOI: 10.3115/v1/D14-1162.

[81] L. L. Huang and S. Bowman, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP*, 2019, pp. 3980–3990.

[82] R. Agerri, I. Aldabe, M. Cuadros, A. Soroa, and E. Agirre, "Multilingual word sense disambiguation using multilingual bert embeddings," *Computational Linguistics*, 2020.

[83] S. Wu and M. Dredze, "Beto, bentz, becas: The surprising cross-lingual effectiveness of bert," *arXiv preprint arXiv:1904.09077*, 2019.

[84] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[85] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[86] A. Nenkova and K. McKeown, "A survey of text summarization techniques," *Mining Text Data*, pp. 43–76, 2012. DOI: 10.1007/978-1-4614-3223-4_3.

[87] T. Chakraborty and S. Bandyopadhyay, "Identification of reduplication in bengali corpus and their semantic analysis: A rule-based approach," in *Proceedings of the Multiword Expressions: From Theory to Applications (MWE 2010)*, Beijing, China, Aug. 2010, pp. 73–76.

[88] A. Senapati, "A fuzzy system for identifying partial reduplication," *Computación y Sistemas*, vol. 26, no. 1, pp. 81–90, 2022. DOI: 10.13053/CyS-26-1-4154.

[89] M. Garcia, M. G. Salido, and M. Alonso-Ramos, "A comparison of statistical association measures for identifying dependency-based collocations in various languages," in *Proceedings of the Workshop on Multiword Expressions (MWE 2019)*, 2019. DOI: 10.18653/v1/W19-5107.

[90] N. Dash, *A Descriptive Study of Bengali Words*. Cambridge: Cambridge University Press, 2015, ch. 8, pp. 225–251.

[91] J. Shallit, *A Second Course in Formal Languages and Automata Theory*, 1st. New York, NY, USA: Cambridge University Press, 2008.

[92] C. Rubino, "Reduplication: Form, function and distribution," *Studies on Reduplication*, vol. 28, pp. 11–29, 2005.

Signature of the Candidate : Abhijit Barman

(Abhijit Barman)

Date : 30/04/2025