

Enhancing the Quality of Devanagari OCR Output for Konkani, Hindi, Marathi and Nepali with Emphasis on Konkani Language

A THESIS SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE & TECHNOLOGY

GOA BUSINESS SCHOOL

GOA UNIVERSITY



Plamga

By

Teja Kundaikar

Discipline of Computer Science & Technology,

Goa Business School

Goa University

Taleigao Plateau, Goa

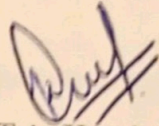
JUNE 2024

DECLARATION

I, Teja Kundaikar, hereby declare that this thesis represents work which has been carried out by me and that it has not been submitted, either in part or full, to any other University or Institution for the award of any research degree.

Taleigao Plateau.

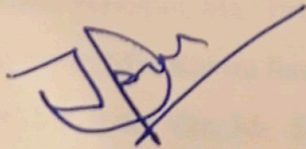
Date: 21-06-2024



Teja Kundaikar

CERTIFICATE

I hereby certify that the work was carried out under my supervision and may be placed for evaluation.



Dr. Jyoti D. Pawar

Professor of Computer Science & Technology,

Goa Business School,

Goa University

Acknowledgment

I am thankful to Prof. J. A. Laxminayaran for introducing me to this research problem during my graduation. I would like to extend my heartfelt thanks to my guide, Prof. Jyoti D. Pawar, for offering me the opportunity to carry out this research under her guidance. I appreciate her valuable suggestions and support in helping me explore this significant path. My heartfelt thanks also go to my DRC members, Dr. Veena Thenkanidiyoor and Shri. Ramdas Nagesh Karmali, for providing valuable insights. I express my gratitude to the Visvesvaraya PhD Scheme for providing financial assistance.

My sincere gratitude goes to Prof. Vineet Chaitanya from IIIT-Hyderabad for introducing me to the OCR alignment process and allowing me to work on it during the Summer School organized by IIIT-Hyderabad. I would like to express my gratitude to the anonymous reviewers for their constructive feedback.

I thank Mr. Ramrao Wagh and Mr. Chinmay Ghaisas for providing Konkani newspaper clipping. I thank Mr. Rahul Kundaikar for cropping images from Konkani newspaper for testing Konkani OCR performance. I thank Mr. Tejas Lotlikar for helping in developing the interface for Konkani OCR. I thank Mr. Sachin Patil and Mr. Sameer Kundaikar for providing computer-related assistance to conduct the experiment smoothly.

I thank my friends and colleagues Dr. Anthony Arthur A. Viegas, Mr. S. Baskar, Ms. Yma Pinto, Dr. P. Payaswini, Dr. Aniket Goankar, Ms. Preeti Khorjuvekar, Mr. Jarret Stevan Anthony Fernandes, Ms. Vandana Naik, Ms. Pradnya Bhagat, Mr. Venkatesh Prabhu, Ms. Priya Lotlikar, Ms. Mamata Verlekar, Ms. Pratham Kurtikar, Mr. Vishal Kinjavadekar, Mr. Wazid Shah, Mr. Gaurang Bane, Ms. Kavita Sawant, Mr. McNeil Rodrigues, Ms. Shubhada Narvekar, Ms. Shweta Bhosle, Ms. Tina Vaz, Mr. Sandesh Naik, Mr. Akshay Shirodkar, Mr. Pranav Shetgaonkar, Mr. Rajendra Gaonkar, Mr. Aditya Parab, Ms. Vishvaja Faldessai who have been a part of my research journey and providing necessary support throughout my journey. I thank Ms. Manika Bindal, Mr. Hanumant Redkar and Ms. Gaurpriya Chodanker for providing peer feedback and support. I thank Prof. Jyoti D. Pawar, Shri Ramdas Karmali and Mr. Swapnil Fadte, for providing help in proofreading this thesis. I was blessed with the support of the Discipline of Computer Science & Technology, Goa Business School, Goa University.

I thank our Dean, Prof. Jyoti D. Pawar, Vice-Deans, Prof. Nilesh Borde, Prof. Pranab Mukhopadhyay, and former Deans Prof. Venkatesh V. Kamat, Prof. M. S. Dayanand, former Vice-Deans, Prof. P.K. Sudarsan and Prof. K.B. Subhash for providing smooth administrative support and guidance.

I am grateful to my family for allowing and supporting me to undergo this valuable process of research. I thank Chandrakant, Amita Kunkolkar and Rajesh Kundaikar for providing support to their best. I thank my siblings Soniya, Rahul, Mandar, Meena, and husband Swapnil, who believe in the richness of learning and providing support. I am fortunate to have friends Ms. Sabella Periera and Dr. Atul Kumar Ojha, who were always there for me, even from a distance. I thank my friends Mamta, Mangesh, Anish, Ritesh, Vivek and Nitesh for all their support. I am blessed with little hearts, Selena and Swatej, for making this journey a cheerful one.

I sincerely extend my heartfelt thanks to the anonymous reviewers for reviewing this thesis and providing detailed and thoughtful suggestions, which contributed to my learning.

तुंका देव बरें करूं.

This thesis is dedicated to wonderful parents

Chudu and Sunita Kundaikar

who raised me to be the person I am today.

Abstract

India is a multilingual country with rich literary traditions. With the advent of the Internet and the digital age, many languages find themselves at a disadvantage. Their continued development and survival is subject to making the transition to digital space. Availability of Natural Language Processing (NLP) tools will play a significant role in making this transition possible. Optical Character Recognition (OCR) is a tool that can help to digitize documents. However, for many Indian languages OCR tools are not available, or demonstrate inadequate accuracy compared to English OCRs (Latin script). Development of OCR for Indian languages poses distinct challenges. Abundance of vowels, consonants, *ligatures* (consonant cluster) and their inflectional nature with complex character graphemes are issues that need attention. Further, the lack of analysis on commonly used fonts, and the absence of standardized test datasets containing ground truth datasets also adds to the challenge. The main objective of this research work is to develop OCR for Konkani language. Konkani is the official language of the state of Goa and one of the 22 official languages of India. It belongs to the Indo-Aryan language family and is primarily spoken in Goa and neighboring coastal regions of Karnataka, Maharashtra, and small pockets of Kerala. There are 12 major scripts widely used for writing Indian languages viz., Bengali, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Meetei, Odia, Ol-Chiki, Perso-Arabic, Tamil, and Telugu, widely used in India. The Devanagari script, which evolved from the Brahmi script, is used by ten Indian languages, Bodo, Dogri, Hindi, Konkani, Maithili, Marathi, Nepali, Sanskrit, Santhali and Sindhi encompassing a large pool of the Indian population who speak and write in these languages. Devanagari is recognized as the official script for Konkani.

The OCRs available for languages using the Devanagari script are Tesseract, Indsenz, e-Aksharayan, and Multilingual Indic OCR. Tesseract OCR (an open source) solution supports Hindi, Marathi, Nepali and Devanagari. Indsenz commercial OCR has support for Hindi, Marathi, and Sanskrit. e-Aksharayan has supports for Hindi. And multilingual OCR for Indic script, which covers

Hindi and Marathi. Multilingual Indic OCR has shown 91.60% word accuracy for Hindi, when tested with data with a single font. Despite the advancements in OCR technology, the accuracy of these systems for Devanagari still needs improvement. Understanding and addressing the challenges is crucial for enhancing the performance and reliability of Devanagari OCR. The usage of Devanagari characters, syllables (akshara), and *ligatures* varies from language to language. Moreover, their appearance differs for different fonts. Performance studies of existing OCRs were conducted on two Hindi datasets with different fonts *Aparajita* and *Nakula* font. It is observed that the OCR accuracy varies for different fonts. To improve the OCR text accuracy, this work has proposed an error correction approach, the Weighted Syllable Spell Correction (WSSC) approach, which handles substitution errors for these two datasets with *Aparajit* and *Nakula*. However, this approach does not handle insertion and deletion errors. To handle insertion and deletion errors, another error correction approach using Masked-Language Modeling with Bidirectional Encoder Representations from Transformers (MLM-BERT) has been proposed to provide contextual suggestions for incorrect words, which has shown improvement in word accuracy. Another observation is that while the suggested word may be contextually appropriate, it might still be incorrect. In some cases, the length of the suggested word can be significantly greater or less than the actual word, which affects the character accuracy. On testing the Hindi OCR on Konkani documents, it is observed that it does not recognize character ँ ([ə]) and consonant with Zero-Width Joiner (ZWJ) (ꣳ [rjə]) consistently. The top 6 errors are due to ँ ([ə]), *anusvara* (ँ), *candra* (ँ), right (’), left (’) single quotation mark, punctuation mark (.), *matra* (ँ) and consonants with ZWJ (ꣳ [rjə]). There were also errors due to syllables and *ligatures*. In order to enhance the OCR accuracy, further an error correction a hybrid approach, using the *MLM-BERT* and *Norvig Spell-Corrector (MB-NSC)* is proposed, which provides suggestions based on context along with the nearest likely word. MB-NSC was tested for Konkani, which has further enhanced the OCR output word and character accuracy by handling substitution and insertion errors effectively.

This work proposes a solution for Konkani OCR by building on Hindi OCR for Konkani documents. Creating a dataset to test OCR for Konkani was essential as no Konkani dataset was available. Accordingly, datasets for Konkani OCR and OCR post-processing performance study are created for a single *Nakula* font. In this work, during OCR post-processing in the initial phase, error detection was performed using a dictionary-based approach on Konkani and Hindi separately. Here, the question arises, can we have a unified error detection approach for lan-

guages written in Devanagari? To work on this, four languages were selected, viz., Konkani, Hindi, Marathi and Nepali. The individual OCR datasets were created using Indian Languages Corpora Initiative (ILCI) data for these four languages. The properties of these datasets were verified with the Hindi dataset used for testing Multilingual Indic OCR. Initially, the performance of existing OCR systems on these four languages was studied separately, and the best OCR output (i.e., Tesseract OCR output) was selected for further error detection work. Accordingly, the error detection approach was proposed using Multilingual BERT (mBERT) and CNN. A dataset for training and testing was created for the error detection approach using the proposed *text2img-doc* and *post-OCR-align* procedures. The proposed error detection approach using mBERT and CNN, creating a Devanagari multilingual model, proved to be beneficial for four Indian languages namely Konkani, Hindi, Marathi, and Nepali. This work studies the percentage of type of errors of existing OCR on different fonts *Nakula*, *Baloo*, *Dekko*, *Briyani* and *Aparajita*. It was observed that types of OCR errors are not consistent for different fonts. This work has contributed to the design of multi-font OCR for Hindi, supporting five fonts using LSTM (open-source OCRopus), which showed negligible deletion errors. This research work also addresses the need for a multi-font OCR for Konkani. Although Devanagari Tesseract OCR exists, studying its performance on Konkani documents was essential; thereby, a multi-font Konkani dataset was created. Testing the existing Devanagari OCR on the multi-font Konkani dataset covering ten fonts revealed that its accuracy varied across different fonts, with the top most two frequent errors involving consonants. A multi-font OCR for Konkani has been designed, using the Devanagari Tesseract OCR as the base model. This Konkani OCR achieved an average word accuracy of 95.94% across ten fonts. Konkani OCR is hosted and currently made available on the Goa University private network for testing purposes.

Contents

Abstract	i
List of Tables	xi
List of Figures	xvii
List of Abbreviations	xxi
1 Introduction	1
1.1 Background	1
1.2 Importance of Fonts in OCR	3
1.3 Motivation	5
1.4 Research Contributions	8
1.5 Thesis Outline	10
2 Related Work	13
2.1 Work on OCR and Post-Processing for Indian languages	13
2.2 OCR for Indian languages	13
2.2.1 OCR using segmentation based approaches	14
2.2.2 OCR using segmentation free approaches	14
2.3 OCR post-processing approaches for Indian languages	16
2.3.1 OCR error detection approaches	16
2.3.2 OCR error correction approaches	17
2.4 Research Gaps	19
3 Empirical study of Devanagari characters, <i>ligatures</i> and syllables: Konkani, Hindi, Marathi and Nepali.	21

3.1	Background	21
3.2	Motivation	23
3.3	Data (Corpus) used	23
3.4	Procedure to compute frequency of characters, syllables (<i>ligature-vowels-signs-diacritic</i>) and <i>ligatures</i> from text corpus	24
3.5	Devanagari Characters and their frequency	25
3.5.1	Independent Vowels	26
3.5.2	Consonants	30
3.5.3	Dependent Vowels and various Devanagari signs	31
3.5.3.1	Dependent Vowels	31
3.5.3.2	Devanagari Signs	32
3.5.4	Additional Consonants and Vowels	33
3.5.5	Punctuation marks and others	35
3.5.6	<i>Viram/Halant</i>	36
3.6	Devanagari syllables (<i>ligature-vowels-signs-diacritic</i>) and their frequency	37
3.7	Devanagari <i>ligatures</i> and their frequency	37
3.8	Observation and Discussion	38
3.9	Conclusion	39
4	Datasets Created for OCR Performance study	41
4.1	Background and Motivation	41
4.2	Using Manual Procedure	42
4.2.1	Hindi datasets for <i>Aparajita</i> and <i>Nakula</i> fonts.	42
4.2.1.1	Characteristics of data	42
4.2.1.2	<i>BM_Aparajita</i> and <i>BM_Nakula</i>	44
4.3	Using <i>text2img-doc</i> and <i>post-OCR-align</i> procedure	44
4.3.1	Procedure <i>text2img-doc</i> for OCR.	44
4.3.2	Procedure <i>post-OCR-align</i> for OCR post processing	45
4.3.3	Hindi datasets for OCR post-processing.	45
4.3.3.1	Characteristics of data	45
4.3.3.2	<i>Hindi-IIITH-dataset</i> and <i>Hin_test_dataset</i>	46
4.3.4	Konkani datasets for OCR and OCR post-processing	46

4.3.4.1	Characteristics of data	47
4.3.4.2	<i>Konkani_OCR_large_test_dataset</i>	50
4.3.4.3	<i>Konkani_OCR_subset_test_dataset</i>	50
4.3.4.4	<i>Konkani_EC_test_dataset</i>	51
4.3.5	Konkani dataset for Multi-font to test OCR (<i>Konkani_test_dataset</i>) . . .	51
4.3.6	Indic datasets for OCR and OCR post-processing	52
4.3.6.1	<i>Indic-OCR-test-dataset</i>	52
4.3.6.2	<i>Indic-post-OCR-dataset</i>	53
4.4	Conclusion	54
5	Algorithms Proposed and Implemented for OCR Error Correction	55
5.1	Background and Motivation	55
5.2	Evaluation Metrics	56
5.3	Performance of OCRs for printed text in different font	57
5.4	Error Correction with Weighted Syllable based Spell Correction (WSSC) . . .	58
5.4.1	Structure of proposed solution	58
5.4.2	Creation of Frequency based Error Mapping Tuple (<i>SY_ERR_MAP</i>) . .	58
5.4.3	Weighted Syllable based Spell Correction Approach	60
5.4.3.1	Error Detection	60
5.4.3.2	Error Correction	61
5.4.3.3	Words Suggestion using error mapping tuple <i>SY_ERR_MAP</i> .	61
5.4.3.4	Best Word Suggestion	63
5.4.4	Results and Analysis	63
5.5	Error Correction using Masked-Language Modeling With Bidirectional Encoder Representations from Transformers (MLM-BERT)	67
5.5.1	Structure of proposed solution	67
5.5.2	Datasets	67
5.5.3	Experimental setup for Error correction using MLM-BERT	68
5.5.4	Results and Analysis	71
5.6	Error correction with the hybrid approach: MLM-BERT and Norvig Spell Cor- rector (MB-NSC)	75
5.6.1	Motivation	76

5.6.2	Structure of proposed solution	78
5.6.3	Experimental setup to study the performance of Tesseract Hindi OCR on <i>Konkani_OCR_large_test_dataset</i>	79
5.6.4	Experimental setup for proposed OCR error correction approach:MB-NSC	79
5.6.4.1	Datasets	80
5.6.4.2	Error correction using MLM BERT for Konkani	81
5.6.4.3	Error correction using Peter Norvig spell corrector for Konkani	82
5.6.4.4	Error correction by proposed MB-NSC hybrid approach . .	83
5.6.5	Results and Analysis:Performance of Tesseract Hindi OCR on Konkani documents	84
5.6.5.1	Performance of Tesseract Hindi OCR on <i>Konkani_OCR_large _test_dataset</i>	85
5.6.5.2	Top 90 characters/ <i>ligatures</i> /syllables that caused errors	85
5.6.5.3	Top 50 words that were incorrectly recognized and words that consistently showed zero accuracy:	88
5.6.6	Result and Analysis of MB-NSC on Konkani data	90
5.6.6.1	Performance of MB-NSC on <i>Konkani_EC_test_dataset</i>	90
5.6.6.2	Error analysis of MB-NSC approach	92
5.7	Conclusion	98
6	Algorithm Proposed and Implemented for Error Detection: Devanagari Multilin- gual Model	101
6.1	Background and Motivation	101
6.2	Structure of proposed solution	103
6.3	Datasets	104
6.4	Performance of Hindi OCR on languages:Konkani, Hindi, Marathi and Nepali	104
6.5	Experimental setup for Error Detection using mBERT	105
6.6	Results and Analysis	109
6.7	Conclusion	114
7	Multi-Font Text Recognizer for Hindi	117
7.1	Background and Motivation	117
7.2	Data creation for text line in five fonts	117

7.3	Performance of existing OCRs	119
7.4	Experimental setup for Multi-Font Text Recognizer using LSTM neural networks	120
7.5	Results and Analysis	120
7.5.1	Accuracy of OCR's on text document images of different font	120
7.5.2	Error analysis for Tesseract OCR, Indsenz OCR, e-Aksharayan OCR and MFD_OCR for each test dataset.	121
7.5.3	Percentage of types of errors for each of the OCR's for different types of fonts style	123
7.6	Conclusion	123
8	Multi-Font OCR for Konkani	127
8.1	Background and Motivation	127
8.2	Datasets	129
8.3	Performance of Devanagari Tesseract OCR on Konkani test data having 10 fonts <i>Konkani_test_dataset</i>	130
8.4	Structure of proposed solution	132
8.5	Experimental set up to create OCR for Konkani using Tesseract LSTM	132
8.6	Results and Discussion	134
8.6.1	Performance of <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_data</i>	135
8.6.2	Error analysis of <i>Konkani_Multi_Font</i> OCR	136
8.6.2.1	Type of OCR errors: insertion substitution and deletion	137
8.6.2.2	Type of substitution errors	137
8.6.3	Comparison of top 10 substitution errors exhibited by Devanagari Tesser- act OCR and <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i> for <i>Category-I</i> fonts	141
8.6.4	Performance of <i>Konkani_Multi_Font</i> OCR on real data	142
8.7	Konkani OCR interface	143
8.8	Conclusion	143
9	Conclusion	147
9.1	Contributions	147
9.2	Summary	149
9.3	Scope for Future Research	153

Appendix A Examples of Devanagari characters, <i>ligatures</i> and syllables: Konkani, Hindi, Marathi and Nepali	155
A.1 Independent vowels	155
A.2 Consonants	157
A.3 Dependent Vowels	160
A.4 Some example of syllables (<i>ligature-vowels-signs-diacritic</i>) and its frequency .	160
A.5 Some example of <i>ligatures</i> whose occurrence is zero in one of the languages . .	165
Appendix B Example of errors introduced due to use of Hindi OCR on Konkani documents	167
B.1 Top 90 syllables contributed to errors along with its frequency, error % and top 6 OCR recognized substitutions.	167
B.2 Top 50 Konkani words misrecognized by Tesseract OCR	174
Appendix C Example of word and syllables correctly and incorrectly recognized by Konkani_Multi_Font OCR	175
C.1 Most frequent words recognized correctly by <i>Konkani_Multi_Font</i> OCR for all ten fonts (i.e. <i>Amiko, Lohit Devanagari, Biryani, Nakula, Cambay, Gargi, Asar and Baloo</i> and <i>Aparajita</i>)	175
C.2 Substitution errors exhibited by <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i> for <i>Category-I</i> (i.e. <i>Amiko, Lohit Devanagari, Biryani, Nakula, Cambay, Gargi and Asar</i>) fonts	176
C.3 Substitution errors exhibited by <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i> for <i>Category-II</i> (<i>Baloo</i> and <i>Aparajita</i>) fonts	178
C.4 Words exhibit zero accuracy for all ten fonts (i.e. <i>Amiko, Lohit Devanagari, Biryani, Nakula, Cambay, Gargi, Asar, Baloo</i> and <i>Aparajita</i> fonts)	180
References	183
List of Publications/Presentations	191

List of Tables

2.1	Summary of related work of Devanagari OCRs built using deep learning	15
3.1	Data used to study characteristics of Konkani, Hindi, Marathi, and Nepali . . .	23
3.2	Independent vowels and their frequency in Konkani, Hindi, Marathi and Nepali	27
3.3	Consonants and their frequency in Konkani, Hindi, Marathi and Nepali	30
3.4	Dependent Vowels Sign and their frequency in Konkani, Hindi, Marathi and Nepali	33
3.5	Devanagari signs and their frequency in Konkani, Hindi, Marathi and Nepali . .	34
3.6	Additional Consonants & Vowels for Sanskrit and their frequency in Konkani, Hindi, Marathi and Nepali.	35
3.7	Generic punctuation and their frequency in Konkani, Hindi, Marathi and Nepali	36
3.8	<i>Viram</i> sign and its frequency in Konkani, Hindi, Marathi and Nepali	36
3.9	Example of syllables (<i>ligatures-vowels-signs-diacritic</i>) and their frequency in Konkani, Hindi, Marathi and Nepali	37
3.10	Some example of <i>ligatures</i> whose occurrence is zero in one of the languages . .	38
3.11	Binary Jaccard Similarity score for Devanagari characters, syllables and <i>liga-</i> <i>tures</i> between each of the languages: Konkani, Hindi, Marathi and Nepali . . .	39
4.1	Characteristics of available Hindi data to test OCR	43
4.2	Characteristics of data used for <i>Hin_test_dataset</i> and <i>Hindi-IIITH-dataset</i> to test error correction approach	46
4.3	Properties of gold standard Konkani raw text corpus from LDC-IL	47
4.4	Category wise top 10 frequent word appearing in <i>Konkani_OCR_large_test_dataset</i>	48
4.5	Top 10 bigrams of the words and its frequency from Konkani corpus LD-CIL (Ramamoorthy et al., 2019)	49

4.6	Top 10 bigrams of the syllables and its frequency from Konkani corpus LD-CIL (Ramamoorthy et al., 2019)	50
4.7	Konkani text in ten fonts	51
4.8	Characteristics of data used to create <i>Indic-OCR-test-dataset</i>	52
4.9	Data used for creating <i>Indic-post-OCR-dataset</i>	53
4.10	Sample text in one of the file in the <i>Indic-post-OCR-dataset</i>	53
5.1	OCRs Word Accuracy for <i>Aparajita</i> and <i>Nakula</i> fonts.	57
5.2	Example of Substitution, Insertion and Deletion errors.	58
5.3	Terminology and Notation used	59
5.4	Creation of Frequency-based Error Mapping Tuple (<i>SY_ERR_MAP</i>)	59
5.5	An example of correct word and OCRed word (output of Tesseract OCR).	59
5.6	Word accuracy obtained using WSSC	65
5.7	Output of WSSC	66
5.8	Output of <i>Hindi_Error_Correction_Model</i> indicating suitable output as per context.	66
5.9	Training Data used to create <i>Dev-Hindi</i> tokenizer and <i>Hindi_Error_Correction_Model</i>	68
5.10	Example of <i>Hindi_Error_Correction_Model</i> output Hindi language.	71
5.11	Word accuracy of Tesseract OCR and <i>Hindi_Error_Correction_Model</i>	72
5.12	Word error rate(WER) of <i>Hindi_Error_Correction_Model</i> testing on <i>Hin_test_dataset</i> and <i>Hindi-IIITH-dataset</i> for each Category of words	73
5.13	An example of correct suggestion provided by <i>Hindi_Error_Correction_Model</i>	73
5.14	Output of <i>Hindi_Error_Correction_Model</i> indicating suitable output as per context.	74
5.15	Output of <i>Hindi_Error_Correction_Model</i> suggest correct suggestions for Noun.	75
5.16	Output of <i>Hindi_Error_Correction_Model</i> having quite similar suggestion to actual word or appropriate based on context or word.	76
5.17	Output of <i>Hindi_Error_Correction_Model</i> Although incorrect suggestion, but valid based on context.	76
5.18	Output of <i>Hindi_Error_Correction_Model</i> provided no suggestion	77
5.19	More example of Output of <i>Hindi_Error_Correction_Model</i> providing No suggestion.	77

5.20	Output of <i>Hindi_Error_Correction_Model</i> with incorrect words.	77
5.21	Training and Test data used for Error Correction approach (MB-NSC).	81
5.22	Example of <i>K_MLM-BERT</i> output for Konkani.	82
5.23	Word Error Rate (WER) and Character Error Rate (CER) of Tesseract Hindi OCR on <i>Konkani_OCR_large_test_dataset</i>	85
5.24	An example words having Zero Width Joiner (ZWJ) and Zero Width Non-Joiner (ZWNJ)	89
5.25	Word accuracy (WA) and Character accuracy (CA) of <i>K_MLM-BERT</i> and <i>Norvig_SC</i> and MB-NSC approach on <i>Konkani_EC_test_dataset</i>	90
5.26	Insertion, substitution, deletion error rate and Character Error Rate (CER) on <i>Konkani_EC_test_dataset</i>	91
5.27	Word error rate (WER) of Tesseract Hindi OCR and MB-NSC approach for each category of words	91
5.28	Output of <i>K_MLM-BERT</i> indicating suitable output as per context.	93
5.29	Output of <i>K_MLM-BERT</i> , <i>Norvig_SC</i> and MB-NSC approach	94
5.30	Use cases tested with proposed MB-NSC approach	95
5.31	Words corrected by proposed MB-NSC approach having character ँ, along with its frequency occurring in Konkani corpus	96
5.32	Examples of words having additional <i>Anusvara</i> that are corrected by MB-NSC approach	96
5.33	Incorrect word suggested by <i>Konkani_Error_Correction_MLM-BERT(K_MLM- BERT)</i> and <i>Norvig_SC</i>	97
6.1	Actual text in Konkani and its OCR recognized output	102
6.2	Characteristics of <i>Indic-post-OCR-dataset</i>	104
6.3	Performance of Tesseract and Indsenz OCR using <i>Indic-OCR-test-dataset</i>	104
6.4	Multilingual BERT Tokenizer output	107
6.5	TP, TN, FN, and FP of error detection models tested with each of the languages	110
6.6	Precision, Recall and F-score for error detection models	110
6.7	Example of <i>Dev-multi-lingual-Error-Detection-Model</i> output for each of the lan- guages.	111
6.8	<i>Dev-multi-lingual-Error-Detection-Model</i> output correctly classify context words (real errors).	112

6.9	Output of <i>Dev-multi-lingual-Error-Detection-Model</i> for 3 types of words 1) Interjections 2) Valid words having similar spelling 3) Words having Canonical Equivalence character	113
6.10	Output of <i>Dev-multi-lingual-Error-Detection-Model</i> for sentence having code mixed (Konkani & Hindi).	113
6.11	Output of error detection <i>Dev-multi-lingual-Error-Detection-Model</i> for Nepali.	114
7.1	Performance of OCR's Character Accuracy for text in different font.	119
7.2	Error analysis of OCR's for text in Aparajita font.	119
7.3	Error analysis of OCR's for text in <i>Nakula</i> font.	119
7.4	OCR's Character Accuracy on test dataset	121
7.5	Percentage of OCR's errors for BMT_ <i>Nakula</i> test dataset.	121
7.6	Percentage of OCR's errors for BMT_ <i>Baloo</i> test dataset.	122
7.7	Percentage of OCR's errors for BMT_ <i>Dekko</i> test dataset.	122
7.8	Percentage of OCR's errors for BMT_ <i>Biryani</i> test dataset.	122
7.9	Percentage of OCR's errors for BMT_ <i>Aparajita</i> test dataset.	123
8.1	Sample Konkani text in ten fonts	130
8.2	Performance of Devanagari Tesseract OCR on multiple fonts	131
8.3	Top 10 substitution errors exhibited by Devanagari Tesseract OCR on <i>Konkani_test_dataset</i> for <i>Amiko, Biryani, Nakula, Cambay, Eczar, Gargi</i> and <i>Asar</i> fonts	132
8.4	Performance of <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i>	135
8.5	Output of Devanagari Tesseract OCR and <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i> for <i>Amiko</i> font	136
8.6	Percentage of insertion, substitution and deletion errors exhibited by <i>Konkani_Multi_Font</i> OCR	138
8.7	Average percentage of errors exhibited by <i>Konkani_test_dataset</i> for <i>Category-I</i> and <i>Category-II</i> fonts for each type of substitution error	138
8.8	Top 10 substitution errors exhibited by Devanagari Tesseract OCR and <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i> for <i>Category-I</i> fonts	141
8.9	Words from newspaper clip in different fonts and its corresponding Devanagari Tesseract OCR and <i>Konkani_Multi_Font</i> OCR output	143
A.1	Examples of Independent vowels in Konkani, Hindi, Marathi and Nepali	156

A.2	Examples of consonants in Konkani, Hindi, Marathi and Nepali	159
A.3	Dependent vowel Sign examples in Konkani, Hindi, Marathi and Nepali	160
A.4	Example of syllables (<i>ligature-vowels-signs-diacritic</i>) and their frequency in Konkani, Hindi, Marathi and Nepali	165
A.5	Some example of <i>ligatures</i> whose occurrence is zero in one of the languages . .	166
B.1	Top 90 syllables contributed to errors along with its frequency, error % and top 6 OCR recognized substitutions.	173
B.2	Top 50 frequently misrecognized actual Words, their frequency and accuracy in percentage.	174
C.1	Most frequent words which are always recognised correctly by <i>Konkani_Multi_Font</i> OCR on the <i>Konkani_test_dataset</i> for all ten fonts	176
C.2	Substitution errors exhibited by <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i> for <i>Category-I</i> fonts (i.e. <i>Amiko</i> , <i>Lohit Devanagari</i> , <i>Biryani</i> , <i>Nakula</i> , <i>Cambay</i> , <i>Gargi</i> and <i>Asar</i> fonts)	178
C.3	Substitution errors exhibited by <i>Konkani_Multi_Font</i> OCR on <i>Konkani_test_dataset</i> for <i>Category-II</i> fonts (i.e. <i>Aparajita</i> and <i>Baloo</i> fonts)	180
C.4	Words exhibit zero accuracy for all ten font (i.e. <i>Amiko</i> , <i>Lohit Devanagari</i> , <i>Biryani</i> , <i>Nakula</i> , <i>Cambay</i> , <i>Gargi</i> , <i>Asar</i> and <i>Baloo</i> and <i>Aparajita</i> fonts)	182

List of Figures

1.1	Block diagram of OCR process	2
1.2	Text in <i>Nakula</i> font	4
1.3	Text in <i>Baloo</i> font	4
1.4	Text in <i>Dekko</i> font	4
1.5	Text in <i>Biryani</i> font	4
1.6	Text in <i>Aparajita</i> font	5
1.7	A sample multi-font article from Konkani daily newspaper : <i>Bhaangarbhuin</i> , a sample page from <i>Konkani vishwakosh</i> text book and sample word in different fonts	6
1.8	Konkani document having various <i>ligatures</i> (consonant cluster)	7
4.1	A sample text in a different font style a) <i>Aparajita</i> and b) <i>Nakula</i>	42
4.2	Category-wise distribution of total words (3619017 words) in gold standard Konkani LDC-IL corpus	47
5.1	An example for error correction steps in WSSC.	61
5.2	Proposed system of Error correction using MLM-BERT	70
5.3	Proposed architecture of MB-NSC error correction approach	78
5.4	Architecture of Proposed MB-NSC error correction approach	80
6.1	Proposed System: Error Detection Model.	106
6.2	An example for module I and IV in proposed system	106
6.3	Learning Curve for <i>Dev-multi-lingual-Error-Detection-Model</i> . (a) Training and Validation Accuracy. (b) Training and Validation Loss.	109

7.1	Sample text in fonts (from top to bottom): <i>Nakula</i> , <i>Baloo</i> , <i>Dekko</i> , <i>Biryani</i> and <i>Aparajita</i>	118
7.2	Deletion type of errors of OCR's for text document in different fonts	124
7.3	Insertion type of errors of OCR's for text document in different fonts	124
7.4	Substitution type of errors of OCR's for text document in different fonts	125
8.1	<i>Bhaangarbhuin</i> Konkani newspaper articles	128
8.2	Procedure to create <i>kon.traineddata</i> model	133
8.3	Output of Devanagari Tesseract OCR and <i>Konkani_Multi_Font</i> OCR on newspaper clip	142
8.4	Input Konkani book and <i>Konkani_Multi_Font</i> OCR output	144
8.5	Input <i>Bhaangarbhuin</i> newspaper clip and <i>Konkani_Multi_Font</i> OCR output	145
8.6	Konkani OCR tool	146

List of Algorithms

1	Words Suggestion	62
2	Weighted Syllable based Spell Correction Approach (WSSC)	64
3	Best Word Suggestion	65
4	MB-NSC error correction approach	83

List of Abbreviations

BERT	B idirectional E ncoder R epresentations from T ransformers
BLSTM	B idirectional L ong S hort- T erm M emory
CA	C haracter A ccuracy
CBOW	C ontinuous B ag- o f- w ords
C-DAC	C enter for D evelopment of A dvance C omputing
CER	C haracter E rror R ate
CIIL	C entral I nstitute of I ndian L anguages
DPI	D ots P er I ch
DTW	D ynamic T ime W arping
FN	F alse N egative
FP	F alse P ositive
GELU	G aussian e rror l inear u nit
GMM	G aussian M ixture M odel
GSM	G rams per S quare M etre
ICDAR	I nternational C onference on D ocument A nalysis and R ecognition
ILCI	I ndian L anguages C orpora I nitiative
ITRANS	I ndian languages T RANS ~~l~~ iteration
LDC-IL	L inguistic D ata C onsortium for I ndian L anguages
LSH	L ocality-sensitive h ashing
LSTM	L ong S hort- T erm M emory
mBERT	M ultilingual B ERT

MB-NSC	MLM-BERT and Norvig Spell corrector
MLM	Masked-Language Modeling
NER	Named Entity Recognition
NLP	Natural Language Processing
NSP	Next Sentence Prediction
NSC	Norvig Spell Corrector
OCR	Optical Character Recognition
OOV	Out-of-vocabulary
POS	Part-of-speech
RNN	Recurrent Neural Network
SLM	Statistical language models
Sr. No.	Serial Number
TDIL	Technology Development for Indian Languages
TN	True Negative
TP	True Positive
WER	Word Error Rate
WA	Word Accuracy
WSSC	Weighted Syllable based Spell Correction
ZWJ	Zero-Width Joiner
ZWNJ	Zero-Width Non Joiner

Chapter 1

Introduction

1.1 Background

In today's digital era, where information is predominantly stored and shared in digital format, Optical Character Recognition (OCR) plays a pivotal role in digitizing documents and generating corpus. This corpus can facilitate many Natural Language Processing (NLP) applications. OCR solutions are available for many widely spoken languages such as English, French, German, etc. However, for Indian languages, OCR systems have not yet achieved the same success. India has 22 officially recognized languages¹ having rich literature collection. Many are in the form of printed documents and written in various scripts. Mainly 12 scripts are widely used across India: Bengali, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Meetei, Odia, Ol-Chiki, Perso-Arabic, Tamil and Telugu. Indian languages present distinct challenges, which include (i) an abundance of vowels, consonants, *ligatures*, (ii) their inflectional nature with complex character graphemes, (iii) a lack of analysis for commonly used fonts, and (iv) the absence of standardized test datasets containing ground truth data for Indian languages. These factors increase the difficulty of providing accurate OCR solutions for Indian languages..

This work mainly focuses on the languages that use Devanagari script. The Devanagari, which evolved from Brahmi script, is important and widely used in India. Devanagari is used by ten Indian languages: Bodo, Dogri, Hindi, Konkani, Maithili, Marathi, Nepali, Sanskrit, Santhali and Sindhi. Presently, OCRs available for languages written in Devanagari are Tesseract OCR (open-source) (Smith, 2007), which has separate model for Hindi, Marathi, Nepali and Devana-

¹https://www.mha.gov.in/sites/default/files/EighthSchedule_19052017.pdf

gari, the Indsenz OCR (commercial product) has separate OCR for Hindi, Marathi and Sanskrit², e-Aksharayan (tdil, 2018) OCR supports for Hindi and multilingual OCR (Mathew et al., 2016) for Indic script, which covers Hindi and Marathi. Multilingual OCR for Indic script has shown 91.60% word accuracy for Hindi. The accuracy of OCR can be improved using post-processing techniques. OCR post-processing has two phases: (i) error detection and (ii) error correction. Recently error detection approaches using deep learning techniques are LSTM, RNN and GMM. And error correction approaches using deep learning are LSTM (Saluja et al., 2017) and Word embedding and Levenshtein distance (Srigiri and Saha, 2020) have been used for Hindi. However, OCR accuracy still needs improvement with the error correction approaches to reach acceptable level.

Additionally, there is a need to provide OCR solutions for low-resource languages such as Konkani. Konkani, one of the 22 languages recognized in the Indian Constitution, holds the official language status in Goa. It uses Devanagari as the official script for writing. Other than Konkani historically it is also written using Roman, Kannada, and Malayalam. Konkani belongs to the Indo-Aryan language family and is primarily spoken in Goa and neighboring coastal regions of Karnataka, Maharashtra, and small pockets of Kerala. Despite its widespread usage, Konkani encounters challenges in Natural Language Processing (NLP), mainly due to the scarcity of a comprehensive corpus. An OCR for Konkani could assist in resolving this situation. Currently, Konkani needs an OCR solution. Also, no OCR dataset for Konkani is available to test the OCR.

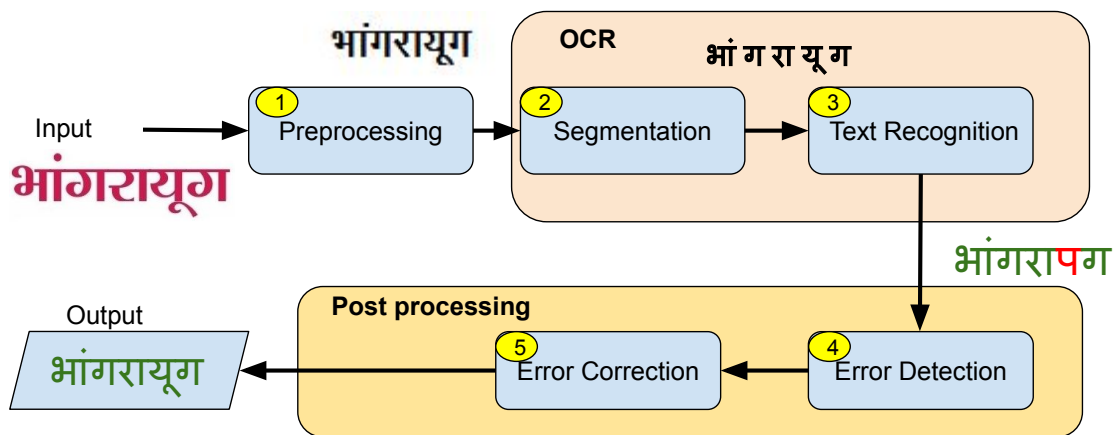


Figure 1.1: Block diagram of OCR process

²<http://www.indsenz.com/int/index.php>

The image text undergoes following five stages during recognition process shown in Figure 1.1.

1. Pre-processing: A scanned document can consist of background noise.. This stage ensures that a good quality image is input into the OCR system.
2. Segmentation: In this phase, the line, word, and characters are segmented. There are two ways to extract character. The first is segmentation-based, and the second is a segmentation-free approach. In the former, the character is segmented from the word. In a segmentation-free approach, a window slides across the text in the image to extract characters and recognize them using deep learning techniques.
3. Text Recognition: In this phase, the text in the image is recognized and made available in an editable text format.
4. Error Detection: OCR recognized text or OCRed text may consist of errors. These are handled by the OCR post-processing phase, which consists of error detection and error correction. Error detection phase detects whether the recognized word is correct (valid) or incorrect (invalid);
5. Error Correction: In this phase, the suitable word is suggested for every incorrect word.

This work mainly focuses on text recognition, error detection, and error correction. .

1.2 Importance of Fonts in OCR

There exist various documents with different fonts. Fonts are a critical factor in the effectiveness of OCR systems. Adequate knowledge of a variety of fonts and understanding font-specific challenges are essential for developing robust and accurate OCR solutions. Font classification is determined by various properties such as internal representation, glyph width, physical existence, installation of font encoding, and readability. The Devanagari font design model (Dalvi, 2019) can visually describe any given Devanagari typeface. The facets of the model consist of tool, grey value, axis, contrast, vertical terminal, horizontal terminal, built, basic hand, angular terminal, counters, turns, vertical stem, inclination, curves, end knot, C to C joinery, beginning loop, end loop, neck join, horizontal proportion, vertical proportion, and neck proportion (Dalvi, 2019). The fonts differ based on the facets chosen for the font design. Over 300 fonts

are available for the Devanagari. Some fonts *Nakula*, *Baloo*, *Dekko*, *Biryani* and *Aparajita*, are discussed forthwith.

Nakula Font The *Nakula* Devanagari font is Unicode compliant and TrueType font. It contains all the conjuncts and other ligatures (including *Vedic accents*) likely needed by Sanskritists. It has rounded glyphs and little thin/thick variation as shown in Figure 1.2

वह कथा या कविता बदल जाती है

Figure 1.2: Text in *Nakula* font

Baloo Font The *Baloo* font is Unicode compliant. It has a distinctive heavy spurless design as shown in Figure 1.3. (Ek-Type, 2019)

वह कथा या कविता बदल जाती है

Figure 1.3: Text in *Baloo* font

Dekko Font The inter-letter spacing of this font is wider, allowing for it to be used at smaller sizes on screens. This font has stroke contrast property. The pen angles traditionally associated with Devanagari has some diagonal stress, but here the script uses a vertical stress as shown in fig 1.4 (Sorkin-Type, 2019)

वह कथा या कविता बदल जाती है

Figure 1.4: Text in *Dekko* font

Biryani Font *Biryani* is a libre font development project. It is designed in a monolinear, geometric sans serif style. Like several early geometric sans typefaces from the last century, *Biryani* font characters have a strong flavor to them. Figure 1.5 shows *Biryani* font in regular font style (font Biryani, 2019).

वह कथा या कविता बदल जाती है

Figure 1.5: Text in *Biryani* font

Aparajita Font *Aparajita* font is high OpenType font and is Unicode-compliant. The Figure 1.6. (Microsoft-corporation, (2019) shows *Aparajita* font in regular font style.

वह कथा या कविता बदल जाती है

Figure 1.6: Text in *Aparajita* font

It is also observed that the *ligatures*³ ल (vertical conjunct³) appearing in word वाचिल्ले ([vatsillē], read) in *Lohit Devanagari* font, and the same word वाचिल्ले ([vatsillē], read) in *Cambay* font shows different appearance. This writing is observed in books, newspapers, etc (refer Figure 1.7).

1.3 Motivation

Devanagari is used by ten languages among the 22 officially recognized Indian languages⁴. The abundance of Devanagari vowels, consonants, and *ligatures*, as well as their complexity, is one of the challenges in developing accurate OCR for these languages. Existing OCRs for languages written in Devanagari, such as Hindi, Marathi, and Nepali, are Tesseract OCR (Source, 2022), Indsenz OCR (Hellwig, 2023), e-Aksharayan OCR (tdil, 2018) and multilingual OCR for Indic scripts (Mathew et al., 2016), where multilingual OCR (Mathew et al., 2016) shows 91.60% word accuracy on Hindi OCR test dataset (Mathew et al., 2016). This test dataset was in a single font, raising questions about OCRs performance for text in different fonts. To study the OCRs performance for text in different fonts. It was essential to create datasets in different fonts.

Efforts have been made to improve the accuracy of Hindi OCRed text using post-processing error correction approaches such as sub-word embedding, LSTM (Vinita et al., 2017; Saluja et al., 2017, 2019), but further improvements in OCR accuracy is desirable. These existing approaches do not suggest words using contextual information. OCR errors are categorized into three types: insertion, deletion, and substitution. Handling insertion and deletion errors using post-processing approaches is challenging. These OCRed words may not be close to the actual word. Enhancing the quality of existing OCR systems can be achieved by employing post-processing approaches that specifically address insertion, deletion, and substitution errors for languages written in the Devanagari. Furthermore, methods that can suggest the correct word

³<https://www.unicode.org/versions/Unicode15.1.0/ch12.pdf>

⁴https://www.mha.gov.in/sites/default/files/EighthSchedule_19052017.pdf

भांगरायूग

Image from text book (konkani vishwakosh)

भाशांतर करतना
अणकारप्याक मूळ संहितेंत
कसलोच बदल करूंक
मेळना. आश्यांत वाड करप
वा तातूंत कसलोय उणाव
करप हाचीय परवानगी ताका
नासता. एके भाशेंतलो
मजकूर दुसरे भाशेंत व्हरप
इतलेंच काम ताणें करपाचें
आसता. लेखनाचें स्वातंत्र्य
हें फकत मूळ लेखकाक
मात तातूंत स्वताची भर
घालपाचें कसलेंच स्वातंत्र्य
भाशांतरकाराक नासता.
मात, एक म्हत्वाची गजाल
म्हणल्यार भाशीक गरजे
प्रमाण कांय संदर्भ, शब्द वा
वाक्यरचना ताणें विचारपूर्वक
करची पडटा.

Image from Konkani Bhaangarbhuin newspaper

थंय आशिल्ल्या यामुनाचार्याक गाळी घाल्यो आनी कर घेवन रोखडोच
कोलाहलाकडेन येवपाक सांगलें. यामुनाचार्याक ताची तिडक आयली.
ताणें म्हळें 'म्हजो गुरू कर भरचो ना. ताचेबदला तुज्या गुरूकडेन
हांव शास्त्रार्थ करपाक तयार आसां'. हें आयकून शिश्यक तिडक
आयली आनी ताणें परत वचून कोलाहलाक घडिल्ली खबर सांगली.
कोलाहलाकूय हें सगळें आयतकच बारा-पंदरा वर्सांच्या
यामुनाचार्याची तिडक आयली. ताणें राजाक ताका आपोवणें धाडपाक
सांगलें.

दुतांकडल्यान विवादाक तयार
आसा म्हणपाचें क पारखातीर
पालखी घाडली

घडिल्ली

Aparajita

घडिल्ली

Nakula

घडिल्ली

Lohit Devanagari

घडिल्ली

थंय आशिल्ल्या यामुनाचार्याक गाळी घाल्यो आनी कर घेवन रोखडोच
कोलाहलाकडेन येवपाक सांगले. यामुनाचार्याक ताची तिडक आयली.
ताणें म्हळें 'म्हजो गुरू कर भरचो ना. ताचेबदला तुज्या गुरूकडेन
हांव शास्त्रार्थ करपाक तयार आसां'. हें आयकून शिश्यक तिडक
आयली आनी ताणें परत वचून कोलाहलाक घडिल्ली खबर सांगली.
कोलाहलाकूय हें सगळें आयतकच बारा-पंदरा वर्सांच्या
यामुनाचार्याची तिडक आयली. ताणें राजाक ताका आपोवणें धाडपाक
सांगलें.

थंय आशिल्ल्या यामुनाचार्याक गाळी घाल्यो आनी कर घेवन रोखडोच
कोलाहलाकडेन येवपाक सांगले. यामुनाचार्याक ताची तिडक आयली.
ताणें म्हळें 'म्हजो गुरू कर भरचो ना. ताचेबदला तुज्या गुरूकडेन
हांव शास्त्रार्थ करपाक तयार आसां'. हें आयकून शिश्यक तिडक
आयली आनी ताणें परत वचून कोलाहलाक घडिल्ली खबर सांगली.
कोलाहलाकूय हें सगळें आयतकच बारा-पंदरा वर्सांच्या
यामुनाचार्याची तिडक आयली. ताणें राजाक ताका आपोवणें धाडपाक
सांगलें.

Figure 1.7: A sample multi-font article from Konkani daily newspaper : *Bhaangarbhuin*, a sample page from *Konkani vishwakosh* text book and sample word in different fonts

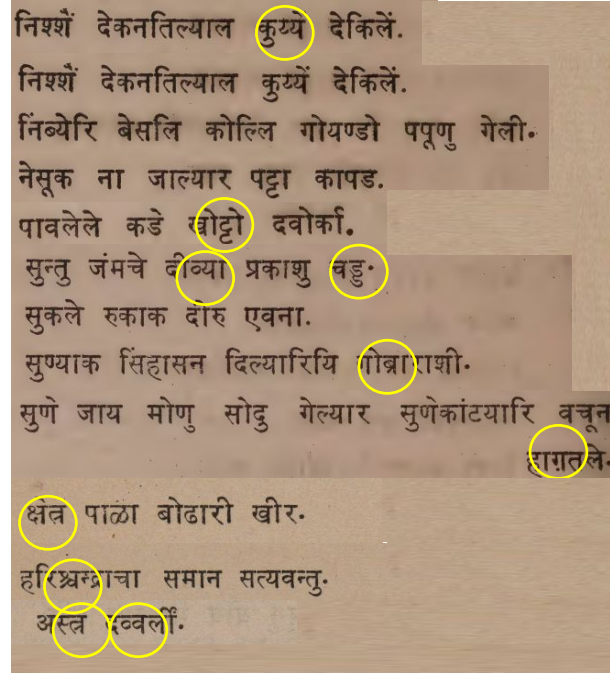


Figure 1.8: Konkani document having various *ligatures* (consonant cluster)

based on context are essential for OCR post-processing, specifically for inflectional languages like Konkani and Marathi.

Developing an OCR for low-resource languages is challenging due to the need for annotated resources. An OCR for low-resource languages like Konkani can be a valuable endeavor to facilitate language-specific NLP applications. Digitizing Konkani books and newspaper articles can be an excellent source for collecting a data corpus. However, the available print text occurs in multiple fonts, as shown in Figure 1.7, a sample clipping from Konkani local newspaper *Bhaangarbhuin*⁵ and Konkani text book. This clipping has text in three fonts. Because of this, having OCR compatibility for several fonts is crucial. Thus, it shows the necessity to develop OCR for Konkani, which supports multiple fonts. One of the solution is to build OCR for Konkani from scratch. But, given low-resource constraint of language like Konkani second approach could be utilizing the existing rich resource OCR supporting Devanagari (i.e. OCR for Hindi). Thus, it was important to study the usage of Devanagari characters, *ligatures*³ and syllables (aksharas⁶)

⁵<https://epaper.thegoan.net/3755632/BHAANGARBHUIN/BHAANGARBHUIN#page/1/1>

⁶Akshara is the fundamental writing unit in Indic scripts, representing an orthographic symbol of a speech sound in Indian languages (Prahallad et al., 2012). Typical form of akshara are V, CV, CCV, CCCV; Akshara follows pattern C*V where C is consonant and V is vowel. Akshara are more like a syllable in English. In this thesis syllable and akshara are used interchangeably.

specific to the language. Yet, no comprehensive study has been done to explore the similarities and differences between Devanagari characters and *ligatures* and syllables (*ligatures* and *ligatures* with dependent vowels and signs) in each of the languages. Figure 1.8 shows example of some *ligatures* appearing in Konkani documents circled in yellow color. Devanagari contains numerous type of *ligatures*. However not all of them may be utilized by all the languages that uses Devanagari. This shows need to study the usage of Devanagari characters, *ligatures* and syllables (*ligatures* and *ligatures* with dependent vowels and signs) in each language. Further, it is important to understand whether a unified post-processing model for error detection can be created. Hence, there is a need to create a dataset to train and test the model, which supports different languages.

1.4 Research Contributions

The research contributions of this work are listed below:

1. **Empirical study of Devanagari characters, *ligatures* and syllables in Konkani, Hindi, Marathi and Nepali.**

Studied the occurrence of Devanagari characters, *ligatures* and syllables in Konkani, Hindi, Marathi and Nepali. It is observed that there exist differences in occurrences of characters, *ligatures* and syllables between each of these languages. Each of the cases is discussed with examples. The binary Jaccard similarity score computation was calculated for each language pair. It is observed that for the Konkani-Marathi pair, the binary Jaccard similarity is 0.41, which is the highest. Hindi resembles more closely to Marathi with a binary Jaccard score of 0.40, while Nepali displays a closer resemblance to Hindi, with a score of 0.31.

2. **Datasets Created for OCR and OCR post-processing performance study.**

Proposed three procedures to create datasets, as listed forthwith (refer 2a), which can be applied to any language with varying fonts and font sizes. Using these procedures, the datasets have been created as mentioned forthwith (refer 2b).

- (a) **Proposed procedures to create dataset**

- i. Manual Procedure adopted to create the dataset.

- ii. Procedure *text2img-doc* for OCR.
- iii. Procedure *post-OCR-align* for OCR post processing.

(b) Datasets Created

- i. Hindi datasets for *Aparajita* and *Nakula* fonts
 - *BM_Aparajita* test dataset
 - *BM_Nakula* test dataset
- ii. Hindi datasets for OCR post processing.
 - *Hin_test_dataset*
 - *Hindi-IIITH-dataset*
- iii. Konkani datasets for OCR and OCR post-processing.
 - *Konkani_OCR_large_test_dataset*
 - *Konkani_OCR_subset_test_dataset*
 - *Konkani_EC_test_dataset*
- iv. Konkani dataset for Multi-Font to test OCR (*Konkani_test_dataset*).
- v. Indic datasets for OCR and OCR post-processing
 - *Indic-OCR-test-dataset*
 - *Indic-post-OCR-dataset* (to train and test OCR post-processing models)

3. Algorithms Proposed and Implemented for OCR Error Correction.

The following three error correction approaches are proposed and implemented for error correction.

- (a) Error correction with Weighted Syllable based Spell Correction (WSSC): Initially studied the performance of existing OCRs on text in two different fonts (i.e. *BM_Aparajita* and *BM_Nakula* dataset). It was observed that OCRs accuracy varies for two different fonts, thereby proposing error correction approach Weighted Syllable based Spell Correction (WSSC). This approach has been tested for Hindi using two OCR data in different fonts (i.e. *BM_Aparajita* and *BM_Nakula* dataset). It handles substitution errors well but can not handle insertion and deletion errors. It is adaptive to new errors.
- (b) Error correction using Masked-Language Modeling with Bidirectional Encoder Representations from Transformers (MLM-BERT): This approach masks the word with

error and provides suggestions based on context. It was tested on Hindi dataset (i.e. *Hindi_test_dataset* and *Hindi-IIITH-dataset*) and showed improvement in word accuracy. However, it may suggest a word having length double or quite less than the actual word, thereby affecting the character accuracy.

- (c) Error correction with the hybrid approach: MLM-BERT and Norvig Spell Corrector (MB-NSC): To achieve main objective i.e. providing OCR solution for Konkani. Initially, this work explored the utilization of Hindi Tesseract OCR on Konkani documents (i.e., *Konkani_OCR_large_test_dataset* 3.6 million words). Further, the OCR'd text accuracy was improved through the proposed MB-NSC approach. It provides suggestions based on context and suggests the nearest word for every incorrect word. It was tested on Konkani data (i.e. *Konkani_EC_test_dataset*) and showed improvement in both word and character accuracy.

4. Algorithm Proposed and Implemented for OCR error detection: Konkani, Hindi, Marathi, Nepali using Multilingual BERT (mBERT).

An error detection approach is proposed using mBERT and CNN for Konkani, Hindi, Marathi, and Nepali. Here, five models are created. Four models are for individual languages, and the fifth one is Devanagari multilingual model, which supports four languages.

- 5. Multi-Font text recognizer for Hindi (MDF_OCR).** A text recognizer for Hindi is created using LSTM neural networks supporting five fonts. It is tested on the dataset with five fonts namely *Nakula*, *Baloo*, *Dekko*, *Biryani* and *Aparajita*. Analyzed the three OCR errors: insertion, deletion, and substitution. It showed consistent insertion and deletion errors for five fonts. The deletion errors were negligible i.e. less than 1.4%.
- 6. Multi-font OCR for Konkani (*Konkani_Multi_Font OCR*).** A multi-font OCR is created for Konkani using the Devanagari Tesseract OCR model. An OCR tool is created for Konkani and made available within Goa University private network for testing purpose (URL: <http://192.168.101.53:3000/>).

1.5 Thesis Outline

The rest of thesis is organized as follows;

- **Chapter 2** provides detailed literature reviews on OCR techniques applied to Indian languages. These are broadly grouped into (i) Segmentation-based approaches and (ii) Segmentation-free approaches. Also, existing OCR post-processing approaches for Indian languages are reviewed specifically for error detection and error correction. Through literature review research gaps were identified.
- **Chapter 3** provides details of Devanagari characters, *ligatures* and syllables used in Konkani, Hindi, Marathi and Nepali. And it is clear that there exist differences of *ligatures* and syllables between each of these four languages. Each of these cases are discussed with examples. Overall binary Jaccard similarity score was calculated to measure the similarity between languages.
- **Chapter 4** discusses the procedures proposed to create the dataset to study the performance of OCRs. Using these procedures *BM_Aparajita*, *BM_Nakula*, *Hindi_test_dataset* and *Hindi-IIITH-dataset* were created for Hindi. Additionally the datasets *Konkani_OCR_large_test_a* (having 3.6 millions words), *Konkani_OCR_subset_test_dataset*, *Konkani_EC_dataset* and *Konkani_test_dataset* (having 10 fonts) were created for Konkani to study OCR performance. Moreover, datasets having four languages: Konkani, Hindi, Marathi and Nepali languages; *Indic-OCR-test-dataset*, *Indic-post-OCR-dataset* were also created.
- **Chapter 5** provides the details of three proposed and implemented approaches for error correction. (a) Weighted Syllable based Spell Correction (WSSC), (b) Error Correction using Masked-Language Modeling With Bidirectional Encoder Representations from Transformers (MLM-BERT) and (c) Error correction with the hybrid approach: MLM-BERT and Norvig Spell Corrector (MB-NSC).
- **Chapter 6** presents the details of proposed and implemented error detection approach using multilingual BERT (mBERT) supporting four languages: Konkani, Hindi, Marathi and Nepali. Five models are created four for individual languages and the fifth one is Devanagari multilingual error detection model.
- **Chapter 7** presents the details of multi-font text recognizer created for Hindi language. A text recognizer is created for Hindi, supporting five fonts using LSTM neural networks. It also shows detailed analysis of OCR errors; insertion, deletion and substitution.

- **Chapter 8** presents the details of multi-font OCR created for Konkani language.
An OCR for Konkani is created using the base model, the Devanagari Tesseract model.
This Konkani OCR is tested on the Konkani dataset having ten fonts (*Konkani_test_dataset*).
A detailed error analysis is presented.
- **Chapter 9** provides the conclusion highlighting the contributions made through this work, summary of the thesis and directions for future work.

Chapter 2

Related Work

2.1 Work on OCR and Post-Processing for Indian languages

Various approaches are explored for OCR and post-processing for Indian languages. The existing work on OCR for Indian languages is categorized as segmentation-based and segmentation-free approaches (refer Section 2.2). Section 2.2.1 discusses the work on OCR for Indian languages using segmentation-based approaches. Section 2.2.2 details the related work on OCR for Indian languages using segmentation-free approaches. Further accuracy of OCR output can be improved by using post-processing approaches. Post-processing has two parts: error detection and correction (refer section 2.3). Section 2.3.1 details the error detection approaches used for Indian languages. Section 2.3.2 covers the error correction approaches used for Indian languages. And the Section 2.4 discusses the identified research gaps.

2.2 OCR for Indian languages

Optical Character Recognition (OCR) is a research field in which the text is recognized from scanned documents. There are two methods to recognize text from the scanned document. The first is segmentation, and the second is a segmentation-free approach. In the former approach, the characters are extracted from the input document using segmentation techniques, and further, these extracted characters are recognized using neural network techniques. In the segmentation-free approach, deep learning techniques are used to recognize the text in each line. A lot of work is reported on OCR for Roman script, and it shows impressive accuracy on various documents. Similar techniques have been used to create OCR for the Devanagari script, but this OCR's

accuracy lags behind the OCR for the Roman script.

2.2.1 OCR using segmentation based approaches

The initial efforts to recognize Devanagari characters in printed documents began in 1970 through segmentation-based approaches (Sinha, 1973; Palit and Chaudhuri, 1995; Pal and Chaudhuri, 1997). Advancing the researchers developed a syntactic pattern analysis system specifically for the Devanagari script (Sinha, 1973). In the 1990s, significant progress was made with the development of the first complete end-to-end OCR system for Devanagari (Palit and Chaudhuri, 1995; Pal and Chaudhuri, 1997). While OCR for Devanagari remained primarily at the research level during that period, it took a significant leap in the early 2000s when the Center for Development of Advanced Computing (C-DAC) India released the first commercial Hindi OCR called *Chitrangan* (Pal and Chaudhuri, 2004). Later, bilingual OCR for Hindi and Telugu was developed in 2003 (Jawahar et al., 2003).

2.2.2 OCR using segmentation free approaches

Recent advancements have included the utilization of a segmentation-free approach for text recognition in printed documents (Mathew and Jawahar, 2022; Karayil et al., 2015; Mathew et al., 2016; Sankaran et al., 2013; Source, 2022). A BLSTM approach was used to recognize text from Hindi documents, showing word error rate (WER) of 8.62% for good quality documents and 22.15% for poor quality documents (Sankaran and Jawahar, 2012). Devanagari Text Recognition using BLSTM which was tested on two datasets *Dataset-1* consisting of 33 books and 1.5M words, and *Dataset-2* consisting of 11 books, 300K words. The results showed a word error rate (WER) of 30.52% and a character error rate (CER) of 7.06% on *Dataset-1*. For *Dataset-2*, the WER was 40.49% and the CER was 12.31% (Sankaran et al., 2013). Accuracy varied for different fonts. The BLSTM networks and Connectionist Temporal Classification (CTC) was also utilized for text recognition in seven Indian languages, namely Hindi, Malayalam, Kannada, Tamil, Telugu, Gurmukhi, and Bengali (Krishnan et al., 2014). It showed CER 1.80% and WER 5.72% for Hindi. Errors were observed due to the font forming different characters, combining two existing characters, numerals, symbols, and punctuation's. Further exploring LSTM approach for printed Devanagari script showing CER ranging from 1.2% to 9%

#	Method	Test data	Error rate	Limitation
1	Multilingual OCR for Indic Scripts (Mathew et al., 2016)			
	RNN	Hindi OCR test dataset	overall WER 8.4% and WER 6 to 22% for Matras	Detecting the Indic Script prior to recognition
2	Segmentation-free approach for printed Devanagari script recognition (Karayil et al., 2015)			
	LSTM	Deva-DB	CER 1.2 to 9%	Low recognition rate for multi fonts
3	Towards a Robust OCR System for Indic Scripts (Krishnan et al., 2014)			
	BLSTM and CTC	7 Hindi Books having 390K words	CER 1.80% and WER 5.72% for Hindi.	Errors due to fonts forming new valid symbols. Other errors due to numerals, symbols such as commas, quotes, and punctuation.
4	Devanagari Text Recognition: A Transcription Based Formulation (Sankaran et al., 2013)			
	BLSTM	<i>Dataset-1</i> : 33 books with 1.5M words. <i>Dataset-2</i> : 11 books, with 300K words	<i>Dataset-1</i> : CER 7.06%, WER 30.52% <i>Dataset-2</i> : CER 12.31%, WER 40.49%	Accuracy varied for different font and difficulty in recognising punctuation marks and symbols
5	BLSTM Neural Network for Hindi (Sankaran and Jawahar, 2012)			
	BLSTM	testing data: 67K words Hindi data	WER 8.62 to 22.15% CER 5.65 to 15.13%	Incorrect recognition for degraded images

Table 2.1: Summary of related work of Devanagari OCRs built using deep learning

for Deva-DB dataset ([Karayil et al., 2015](#)). However it showed low recognition rate for multi-font. Exploring RNN to build multilingual OCR for Indic script, showed WER 8.4% on Hindi dataset ([Mathew et al., 2016](#)). The test dataset for Hindi was in single font. These segmentation free approach summary is presented in Table 2.1. The limitation of these OCRs is that it showed low recognition for multi fonts ([Karayil et al., 2015](#); [Sankaran and Jawahar, 2012](#)) as mentioned

in Table 2.1. There has been work for different scripts and little work for text document images in multi-font for same scripts. OCR test dataset was available only for Hindi language.

Presently, there are few OCR options for Hindi, including open source Tesseract OCR (Smith, 2007) which supports Hindi, Marathi, Nepali and Devanagari. Indsenz OCR (a commercial product)² which supports Hindi, Marathi, Sanskrit and Gujarati, and Multilingual OCR for Indian languages which support Hindi, Assamese, Bengali, Gujarati, Gurmukhi, Kannada, Malayalam, Manipuri, Marathi, Odia, Tamil, Telugu and English (Mathew et al., 2016). e-Aksharayan (tdil, 2018) OCR which supports Hindi, Kannada, Malayalam, Tamil, Punjabi, Assamese and Bangla. Currently, Konkani language lacks OCR solution. It is essential to study the performance of existing OCR on different fonts. It is also important to study whether OCR's of existing rich resource language can be utilized for low-resource Konkani.

2.3 OCR post-processing approaches for Indian languages

OCR post-processing is a crucial phase in improving OCR accuracy, and it has two phases: error detection and error correction. OCR output is input to this error detection phase. The error detection phase identifies if the word is correct (valid) or incorrect (invalid). If a word is incorrect, it is given as input to the error correction phase. Post-processing is challenging for Indian languages as these languages are highly inflectional. The lack of language tools like morphological analyzers and POS taggers creates difficulties. The abundance of words in the Indian language is another main concern (Vinitha and Jawahar, 2016). Significant work has been done to identify OCR errors and correct them in Indian languages.

2.3.1 OCR error detection approaches

Initially, the dictionary approach was employed to detect OCR errors. This method yields satisfactory results when the word is present in the dictionary. However, it encounters difficulties with out-of-vocabulary words and fails to identify context-based real-word errors within the language (Kukich, 1992). An extensive review of the techniques employed in error detection is provided in the survey (Kumar, 2016; Nguyen et al., 2021).

Literature shows the use of a morphological analyzer to find the root word to detect errors in OCR text (Pal et al., 2000). Additionally, the use of a graph-based, sub-character level language model for Malayalam to detect and correct errors in words (Mohan and Jawahar, 2010) was reported. Shape-based post-processing of the corpus was carried out, and the corpus was categorized based on the size and shape of the word (Lehal et al., 2001). Recently deep learning approaches, such as RNN and GMM has been used to detect OCR errors in three languages, Hindi, Malayalam, and Gujarati (Vinitha and Jawahar, 2016). Further lookup dictionary and context-based Named Entity Recognition (NER) were used to detect possible spelling errors in Hindi text (Pal and Mustafi, 2020). However, despite the advancements, some limitations and challenges still need to be addressed. Some of the common shortcomings of OCR error detection for Indian languages:

- i. Some languages face the unavailability of language tools such as NER, POS taggers, and morphological analyzers. These tools are not readily accessible or developed for every language, particularly for those with fewer resources or lower levels of linguistic documentation.
- ii. One of the challenges in applying deep learning techniques is the limited availability of data and resources for specific languages.
- iii. Currently, no unified model has been applied to a specific script regardless of the language. Each language within a script has its customized model.

2.3.2 OCR error correction approaches

Several OCR post-processing error correction approaches are explored (Nguyen et al., 2021). These techniques can be grouped based on usage of (i) Dictionaries or Lexicon (ii) Syntactic or Semantic rules of the language/word morphology, (iii) Statistical Language Models usually represented as n-grams, etc., and (iv) Deep Learning approach. Some of the methods find the best alternative, while other methods find the top-n possible alternatives.

Initially, an attempt was made for OCR error correction based on dictionary (Bansal and Sinha, 2001). However, this method covers limited words, makes it unsuitable for inflectional language such as Konkani, Hindi, Marathi, Nepali etc. .

A Morphological analyzer was used for Bangla (Indian script) to identify the root words and pro-

vide appropriate suggestions (Pal et al., 2000). The corrections made here are based on a quick dictionary lookup method and suggest the precise word in 82.44% of cases. The shape-based post-processing technique was explored for Gurmukhi, resulting in 3% improvement, showing 97.34% character accuracy (Lehal et al., 2001). Here, based on similarity, the consonants were categorized into groups. Each group was assigned a unique number. Accordingly, the input word was encoded. If the code appears in the dictionary, matching function was executed between the input word and words stored under that particular code. In cases where a match has not occurred, structurally similar words to the input are suggested as alternatives. This approach requires assistance in providing suggestions to similar characters.

Locality Sensitive Hashing (LSH) was utilized to create word clusters to improve the OCR accuracy of Telugu language (Rasagna et al., 2009), yielding in 79.12% word accuracy. Here, the word images OCR outputs within a cluster were compared. Thus, improvements were achieved through methods such as Character Majority Voting or Dynamic Time Warping (DTW) Technique. However, this technique is limited in not offering suggestions for unique words that appear less in a cluster.

To rectify OCR word errors in Malayalam, a graph-based, sub-character level language model was applied (Mohan and Jawahar, 2010). This approach represented language information and visual similarities as edges and nodes respectively. The suggested words have an optimal path from source to destination. Thus it achieved 95.0% word accuracy (Mohan and Jawahar, 2010). This approach needs help to provide suggestions for rare words, proper nouns, and foreign words.

Error correction for Indian languages such as Hindi, Punjabi (Gurmukhi script), Telugu, and Malayalam (Vinita et al., 2017) was performed using statistical language models (SLM) and Dictionary-based methods. Thus, errors were improved in ranges between 38% to 66% for the four languages. The shortcoming of this work is that a dictionary and character n-gram alone is ineffective in suggesting the correct word for the inflectional nature of Indian languages.

Sub-word embeddings were utilized for OCR error correction in Hindi, Sanskrit, Kannada, and Malayalam, showing 90.42% word accuracy for Hindi (Saluja et al., 2019). The additional intersection of cluster analysis and error rectification was seen in the Hindi and English documents when comparable terms were clustered for error correction (Das et al., 2019). Contextual suggestion is absent from this method.

Word Embedding and Levenshtein Distance were utilized for spelling correction in OCR-recognized Hindi text (Srigiri and Saha, 2020), which uses the Continuous Bag-of-Words (CBOW) model. It does not consider the order of words in a given context. Thereby, it struggles to provide appropriate suggestions for incorrect words based on context.

Deep learning techniques were recently applied to detect and correct OCR errors in Hindi, Malayalam, Sanskrit, and Kannada. Specifically, an LSTM approach was utilized, which is particularly pertinent when languages are written in different scripts (Saluja et al., 2017). For Hindi, it displayed 92.0% word accuracy. LSTMs can capture dependencies across longer sequences than ordinary RNNs, although they still have a limited context window.

2.4 Research Gaps

The following are the research gaps that are identified through literature review:

- **Need for accurate OCR for Indian languages supporting multi-font:** Indian OCR systems are not yet able to successfully recognize printed document images of varying scripts, quality, and fonts, especially compared to those for European languages. One of the reason is Indian script is complex due to its many vowels, consonants, conjuncts, and intricate character graphemes. Different fonts can significantly alter the appearance of characters, syllables (akshara), and conjunct consonants, adding to the challenge of accurate OCR recognition.
- **Unavailability of standard test dataset:** No standard dataset is publicly available to study OCR systems' performance, except for Hindi, in a single font. However, there is no OCR dataset for multiple fonts to evaluate the performance of existing OCRs across different font. There is no OCR dataset for Konkani to test OCR and post processing approaches. No dataset is currently available to test models that support multiple languages.
- **Need to study the performance of existing OCRs for different fonts** The data is available in different fonts. It is essential to study the performance of the existing OCRs in different fonts.
- **Need multi-font OCR solution for low-resource language Konkani:** Low-resource languages such as Konkani multi-font need OCR solutions.

- **No study is conducted to utilize rich resource language OCR for low resource language such as Konkani:** There are existing OCRs for Hindi, Marathi and Nepali. It is important to explore the utilization of rich resource language Hindi OCR for low resource language Konkani.
- **No study on the Devanagari characters, syllables and *ligatures* used by each of the languages written in Devanagari:** There are various Devanagari characters, vowels, consonants, syllables, consonant with diacritic and *ligatures* (consonant cluster) for each of the languages written in Devanagari.
- **No error detection and correction model for Konkani:** There is need to build OCR error detection and correction model for Konkani. The question arises: can there be a unified Devanagari post-processing model that serves this purpose for multiple languages?
- **Need to study the performance of Devanagari Tesseract OCR model on Konkani documents:** A Devanagari Tesseract OCR model is available. It is essential to understand whether this model can be used for Konkani.

Chapter 3

Empirical study of Devanagari characters, *ligatures* and syllables: Konkani, Hindi, Marathi and Nepali.

3.1 Background

Devanagari is used by ten Indian languages, which are among the 22 officially recognized languages. Devanagari characters consists of various elements like Consonants¹, Independent vowels¹, Dependent vowels¹, various Devanagari signs¹, *Viram/halant*¹ (used for forming conjunct consonants), additional consonants and vowels, vedic tones¹, accent marks¹, punctuation marks¹, and Devanagari syllables¹³. These elements form the foundation of the Devanagari, which accommodates the diverse pronunciation styles and unique ways of representing syllables (akshara) in Indian languages. The combination of consonants, vowels, and diacritics in Devanagari allows precise representation of various sounds and pronunciation subtleties, although not all characters are used by every language. For example, the character ष ([ʃ]) is quite common in Konkani and Marathi, whereas it has rare usage in Hindi.

Further, there are various conjunct consonant or consonant cluster exists in Devanagari. The consonant cluster forms that serve as orthographic abbreviations (*ligatures*)². The *ligatures* can be of form Conjunct Consonant (e.g. षष्ठ [ʃʂ]), Multi-Consonant Conjuncts³ (e.g. लृष्ट [lʃʂ]), Preventing Conjunct Forms³ (e.g. दृष्ट [dʃʂ]), Dead Consonants³ (e.g. षष्ठ [ʃ]). Further conjunct

¹<https://www.unicode.org/charts/PDF/U0900.pdf>

²This *ligatures* abbreviation takes place only in the context of a consonant cluster(*ligatures*)³

consonant has different formations; (a) half-form its combination with the full form; (e.g. श्रद्ध [ʃd̪ə]) (b) vertical conjunct (e.g. ल [lə]) (c) fullyligated conjunct³, in which the components are no longer distinct (e.g. क्ष [kʃə]).³ and (d) Dead consonant with another consonant forming (*repha*)³. (e.g. र्ष [d̪rʃ]). Moreover, these *ligatures* can combine with dependent vowels and Devanagari signs forming syllable (e.g. दीर् [d̪iːr̥], डी [d̪aːr̥], ह्या [ɦja], ली [liː]).⁴ These syllables includes both *ligatures* and *ligatures* along with dependent vowels and Devanagari signs for convenience termed as *ligature-vowels-signs-diacritic*

Currently, character sets for individual languages are available for Konkani⁵, Hindi⁶⁷, Marathi⁸ and Nepali⁹, etc., but there's no comprehensive study detailing the specific Devanagari characters, syllables, *ligatures* utilized in each language. The comparative study of these languages character, *ligatures* and syllables (*ligature-vowels-signs-diacritic*) is essential. This work details the study of characters, *ligatures* and syllables (*ligature-vowels-signs-diacritic*) in Konkani, Hindi, Marathi, and Nepali. These languages are chosen for the study since a large corpus is available for these languages. This study helps to understand the differences and similarities of the Devanagari characters, *ligatures* and *ligature-vowels-signs-diacritic* in Konkani, Hindi, Marathi and Nepali.

This chapter is organized as follows; Section 3.2 details the motivation for this work. Section 3.3 provide details of data used for this study. Section 3.4 discusses the proposed procedure to compute the frequency of characters, *ligatures* and *ligature-vowels-signs-diacritic*. Section 3.5 provides frequency and example for each of the Devanagari characters in Konkani, Hindi, Marathi and Nepali. Section 3.6 provides the frequency of *ligature-vowels-signs-diacritic* in four languages. Section 3.7 provides frequency of *ligatures* in four languages. Section 3.8, discusses binary jaccard similarity score for Independent vowels, Consonants, Dependent vowels, Devanagari signs, *ligature-vowels-signs-diacritic*, *ligatures* and overall characters along with *ligature-vowels-signs-diacritic*. Section 3.9 concludes the chapter.

³<https://www.unicode.org/versions/Unicode15.1.0/ch12.pdf>

⁴zero or more consonant+ (Nukta*)+ virāma sequences followed by a consonant (+Nukta*) followed by a vowel sign and modifier https://w3c.github.io/ilreq/#h_abnf_based_definition_of_indic_syllable

⁵https://en.wikipedia.org/wiki/Konkani_alphabets

⁶<https://omniglot.com/charts/print/hindi.pdf>

⁷<https://www.hinditracks.in/hindi-letters>

⁸https://en.wikipedia.org/wiki/Marathi_language

⁹<https://nepalilanguage.org/alphabet/>

3.2 Motivation

Indian languages, namely Bodo, Dogri, Hindi, Konkani, Maithili, Marathi, Nepali, Sanskrit, Santhali and Sindhi, use the Devanagari. Currently few OCRs are existing for Hindi, Nepali, Marathi (refer Chapter 2; section 2.2.2). It is important to understand the factors that affects OCR accuracy. OCR accuracy depends on numerous factors and few are listed here specific to Devanagari characters, *ligatures* and syllables. Firstly complexity of script, and Devanagari is intricate. Secondly, the characters, *ligatures* and syllables that every language uses. Third, the characters, syllables and *ligatures* appears differently for various fonts. The characters and syllables appearing in different style based on fonts are discussed in chapter 1; section 1.2. An example of *ligatures*(vertical conjunct) ल्ल [llə] appearing in word वाचिल्लें [vatsillē] which in *Lohit Devanagari* font. And the same word वाचिल्लें [vatsillē] which is in *Cambay* font shows different appearance. This writing is observed in books, newspapers, etc (refer Figure 1.7).

This work investigates the characters, *ligatures*, and syllables (*ligature-vowels-signs-diacritic*) used in Konkani, Hindi, Marathi, and Nepali. This study will help to understand the similarities and differences between these four languages

3.3 Data (Corpus) used

Language	Total Words	Source of data
Konkani	3607600	Konkani dataset (Ramamoorthy et al., 2019)
Hindi	8215774	Hindi dataset (Shivam, 2022)
Marathi	5336244	Marathi dataset (Banerjee et al., 2021)
Nepali	5439759	Nepali dataset(Gaurav, 2019b)

Table 3.1: Data used to study characteristics of Konkani, Hindi, Marathi, and Nepali

The following data is used for this study. The characteristic of data is shown in Table 3.1.

1. **Konkani:** A Konkani raw text corpus ([Ramamoorthy et al., 2019](#)) is taken from Linguistic Data Consortium for Indian Languages (LD-CIL) containing 3607600 words.

2. **Hindi:** Hindi data consisting of 8215774 words taken from huggingface (Shivam, 2022).
3. **Marathi:** Marathi data is used from IIT Bombay consisting of 5336244 words (Banerjee et al., 2021).
4. **Nepali:** Nepali data is taken from Nepali Wikipedia Articles¹⁰ (Gaurav, 2019b) and iNLTK Nepali News Dataset¹¹ (Gaurav, 2019a) consisting of 5439759 words.

3.4 Procedure to compute frequency of characters, syllables (*ligature-vowels-signs-diacritic*) and *ligatures* from text corpus

The following procedure has been used to compute the frequency of the characters, syllables and *ligatures* occurring in text corpus. The procedure steps are shown for Konkani corpus forthwith. The same procedure is followed for other languages.

1. **Data collection in single text file:** Merge all the files from the Konkani corpus and named it as *merge_file.txt*.
2. **Preprocessing of a text file:** For each word *W* in *merge_file.txt* remove single and double quotation mark and brackets, dot (.), numbers, special characters from the text as this work is restricted to Devanagari characters.
3. **Tokenizing text to characters and counting its frequency:** For every character *C* in the word *W*, it is inserted into a data structure that records the character along with its frequency. The frequency is then incremented by 1 for each new occurrence of character. Each of the characters and their frequency appearing in the data structure is written to the file named *Konkani_char_freq.txt*. Thus creating a tuple with character and frequency. (e.g. ॐ, 103942)
4. **Tokenizing the text to syllable (*ligature-vowels-signs-diacritic*) and counting its frequency:** The word *W*, is split in syllable using the python package *unicodedata*. (e.g.

¹⁰<https://www.kaggle.com/datasets/disisbig/nepali-news-dataset>

¹¹<https://www.kaggle.com/datasets/disisbig/nepali-wikipedia-articles>

word धर्मक्षेत्रे [d^hərməkʃet̪re] will be split as 'ध' [d^hə], 'र्म' [rmə], 'क्षे' [kʃe], 'त्रे' [tre]). And each syllable having Devanagari character *halant/viram* is inserted into a data structure that records the syllable along with its frequency. Thus this will only consider 'र्म', 'क्षे', 'त्रे' syllables. The frequency is then incremented by 1 for each new occurrence of a syllable. Each syllable and its frequency appearing in the data structure is written to the file named *Konkani_syllable_freq.txt*. Thus creating a tuple with syllables and their frequency. (e.g. ज्या [dzya], 7587).

5. **Tokenizing text and counting ligatures frequency:** The *Konkani_syllable_freq.txt* is taken as input, and only the *ligatures* is stored from each syllable, trimming the dependent vowel and signs. For example, these are syllable 'र्म' [rmə], 'क्षे' [kʃe], 'त्रे' [tre] as input along with their frequency. From these, only 'र्म' [rmə], 'क्षे' [kʃə], 'त्रे' [trə] *ligatures* will be stored in a data structure along with their frequency. For each new occurrence of ligature, it is inserted in the data structure along with its frequency. Each of the ligature and its frequency appearing in the data structure is written to the file named *Konkani_ligature_freq.txt*. Thus creating a tuple with ligature and its frequency. (e.g. ड [d̪ə], 409).

The above Steps 1, 2, 3, 4, and 5 are repeated for languages Hindi, Marathi, and Nepali. The data for each of the languages is mentioned in section 3.3. Thus initially creating file *Hindi_char_freq.txt*, *Marathi_char_freq.txt* and *Nepali_char_freq.txt* having character frequency for each language. Later creating file *Hindi_syllable_freq.txt*, *Marathi_syllable_freq.txt* and *Nepali_syllable_freq.txt* having syllable frequency for each language. And lastly creating file *Hindi_ligature_freq.txt*, *Marathi_ligature_freq.txt* and *Nepali_ligature_freq.txt* having ligatures frequency for each language.

3.5 Devanagari Characters and their frequency

The Devanagari character set comprises of 7 categories, namely, i) Independent vowels¹ ii) Consonants¹ iii) Dependent vowels¹ and various Devanagari signs¹ iv) Additional consonants and vowels¹ v) Punctuation marks¹ vi) *Viram/halant*¹ (which forms consonant cluster). Characters in each category and their frequency in Konkani, Hindi, Marathi, and Nepali are provided forthwith.

3.5.1 Independent Vowels

Independent vowels are symbols that represent vowel sounds on their own, without being combined with any consonants. In the Devanagari, there are 17 independent vowels¹², detailed in Table 3.2. This table shows the frequency of these independent vowels in four different languages: Konkani, Hindi, Marathi and Nepali. Additionally, Table A.1 (refer Appendix A; Section A.1) presents instances of each independent vowel in four languages, along with their transliterations in ITRANS¹³ and the corresponding gloss. It is observed that eleven independent vowels अ /ə/, आ /a/, इ /i/, ई /i:/, उ /u/, ऊ /u:/, ए /e/, ऐ [æ], औ /ɔ/, ओ /o/ and औ [au] have instances in all the four languages. It is important to note that औ [au] has limited instances in Konkani compared to other three languages. For example औवेस ([auves] 'wow'), औरंगजेब ([auṛgədzɐb] 'Aurangzeb' PRP. N.). Furthermore there are no instances of syllables with canonical equivalence¹³ consisting of two characters, अ (U+0905) and ौ (U+094C) existing in the Konkani gold standard corpus (Ramamoorthy et al., 2019). Furthermore four independent vowels ऋ, ॠ, ऐ, ओ have either rare or no instance in all the four languages. And two vowels ॐ (U+0904) and ॡ (U+090C) do not appear to be in use in all four languages.

Occurrence of the vowel ऋ: Interestingly, the occurrence of the vowel ऋ (U+090B) is significantly low in Hindi as compared to Konkani, Marathi and Nepali. Although, words containing ऋ vowel are borrowed words from Sanskrit. For example, in Konkani ऋतूंकडेन ([ṛtũkəden] gloss: by seasons), ऋशीच्या ([ṛʃitsjə] gloss: of sage). Similarly, in Marathi, ऋ [ṛ] is rarely used and seen in words like ऋषि ([ṛʃi] gloss: sage) and ऋतूमध्ये ([ṛtũmɒd̪hje] gloss: in seasons). However, in Nepali, the vowel ऋ [ṛ] instances are higher as compared to the other three languages. For example words ऋण ([ṛn] gloss: loan), ऋतु ([ṛtu] gloss: season) are Nepali-language specific words. Additionally, it has words such as ऋषिराज ([ṛʃiradz] gloss: Rishiraj), ऋषिमुनी ([ṛʃimuni] gloss: sage), ऋषियों ([ṛʃijõ] gloss: the sages), etc. Also, it is observed that the vowel ऋ has a rare occurrence in Hindi. For example ऋषि का ([ṛʃi ka] gloss: Rishi's), where the action का is separated from noun ऋषि. Thus its occurrence is low as compared to Konkani, Marathi and Nepali.

Occurrence of the vowel ऐ /ɛ/: It is observed that the vowels ऐ (U+090D) and ॠ (U+090E) are

¹²<https://de.ashtangayoga.info/philosophie/sanskrit-und-devanagari/transliterations-tool/>

¹³<https://w3c.github.io/ilreq/>

Sr.	Devanagari Independent Vowel			Frequency			
No.	Character	Unicode	Devanagari Letter	Konkani	Hindi	Marathi	Nepali
1	अ	U+0905	A	103942	259845	208765	205124
2	आ	U+0906	AA	298190	225013	475174	120024
3	इ	U+0907	I	23272	162062	42822	92426
4	ई	U+0908	II	3836	136896	41284	133851
5	उ	U+0909	U	68665	185504	79562	207280
6	ऊ	U+090a	UU	995	10939	25834	2584
7	ऋ	U+090b	VOCALIC R	373	4	599	3990
9	ॠ	U+090d	CANDRA E	138	0	346	10
10	ऐ	U+090e	SHORT E	6	0	1	7
11	ए	U+090f	E	52213	439925	79524	256308
12	ऐ	U+0910	AI	416	11065	6214	3640
13	ऑ	U+0911	CANDRA O	6020	10665	7133	98
14	ओ	U+0912	SHORT O	0	0	5	9
15	ओ	U+0913	O	8270	60968	10185	9070
16	औ	U+0914	AU	174	187714	4498	6955
17	अे	U+0904	SHORT A	0	0	0	0
18	ऌ	U+090c	VOCALIC L	0	0	0	0
19	अँ*	U+0972	CANDRA A	0	0	212	0
* Independent vowel only for Marathi							

Table 3.2: Independent vowels and their frequency in Konkani, Hindi, Marathi and Nepali

not found in Hindi, and their usage is quite limited in Konkani, Marathi and Nepali (see Table A.1 for examples). The vowels ऐ are exemplified as nouns or English words, such as ऐन्थनी ([ɛnt̪həni] 'Anthony' PRP. N.) in Konkani and Marathi. In Nepali the same name *Anthony* is written as ऐन्टोनी [ɛnt̪oni] and in Hindi it is written as ऐथनी [ɛt̪həni]. Thus, the same word is written differently based on how it is pronounced specific to language.

Occurrence of vowel ऐ (U+090E): The vowel ऐ has either rare or no occurrence in all 3 languages. Specifically 6 in Konkani, 1 in Marathi and 7 in Nepali and no occurrence in Hindi. For example, in Konkani ऐशआरामाचेर ([aɪʃəaramatser] gloss: luxurious), ऐवजासयत ([aɪvəjaʃəyt̪]

gloss: with material), ऐरागबाळ्यांचें ([airagəba|jātsẽ] gloss: not neat person), ऐपती ([aipəti] gloss: Apati), ऐवजी ([aivədʒi gloss: instead of) are instances of this vowel. In Marathi only one occurrence having word कोलेस्ट्रमच्याऐवजी ([koləstrmətsjəaivədʒi gloss: instead of colostrum) which is a English word. In Nepali, the occurrence of a vowel ऐ is seen in only two words, i.e. ऐतिहासिक (gloss: historical) and ऐतिहासिकतालार् (aitihāsikətālār] gloss: on historical line). It is observed that letter ऐ is mostly used for borrowed words in the corpus used for this study. And there is no occurrence of this character in Konkani literature. It also could be typing error.

Occurrence of the vowels ओ: The vowel ओ (U+0912) is absent in Konkani and Hindi, and is rarely seen in Marathi and Nepali. For example, in Marathi, ओळखले ([ō|ək^hle] gloss: recognized) is frequently used, but it is actually spelled as ओळखले [o|ək^hle], which appears more frequently in the Marathi corpus used for this study (Banerjee et al., 2021). Similarly, in Nepali, the word ओतप्रोत ([ōtprɔt] gloss: Alarmig) has one occurrence, the same word spelled as ओतप्रोत [otprɔt] has more instances. Thus, indicating ओ is not much used in Marathi and Nepali. ओ vowel is typing error.

Occurrence of the vowels अँ: Additionally, the independent vowel अँ (U+0972), which is exclusively for Marathi¹, is not used in Konkani, Hindi, and Nepali (see Table 3.2). However, its canonical equivalence¹³ exists in Konkani and Hindi having the syllable¹³ अँ, which consists of the two characters अ (U+0905) and ँ (U+0945) appears frequently in Konkani text, with a frequency of 1388, for example, ऑगस्ट [əgəst]. A similar pattern is found in Hindi and Nepali with examples like ऑक्सीजन [əksidʒən], though it's worth noting that these instances are primarily nouns borrowed from English, such as Oxygen and August, etc. In Nepali, Oxygen is written as अक्सिजन [əksidʒən] and August is written as अगस्ट [əgəst]. Mostly character अँ is used to write English words in Devanagari.

Sr. No.	Consonant			Frequency			
	Character	Unicode	Devanagari Letter	Konkani	Hindi	Marathi	Nepali
1	क	U+0915	KA	895828	2427072	1003268	1533406
2	ख	U+0916	KHA	135053	120495	112142	176145
3	ग	U+0917	GA	313617	479835	307540	536663
4	घ	U+0918	GHA	71393	31917	62652	43662
5	ङ	U+0919	NGA	156	0	11	58745

Sr. No.	Consonant			Frequency			
	Character	Unicode	Devanagari Letter	Konkani	Hindi	Marathi	Nepali
6	च	U+091a	CA	539470	239982	603237	211504
7	छ	U+091b	CHA	1955	42146	13840	276219
8	ज	U+091c	JA	310119	540836	347488	364274
9	झ	U+091d	JHA	13368	22767	61318	19848
10	ञ	U+091e	NYA	3362	15709	14821	33081
11	ट	U+091f	TTA	193379	285414	198747	291089
12	ठ	U+0920	TTHA	19872	38346	121794	55589
13	ड	U+0921	DDA	251421	228967	153597	152464
14	ढ	U+0922	DDHA	1848	37949	37607	28495
15	ण	U+0923	NNA	241711	154361	527999	147230
16	त	U+0924	TA	992791	1216018	1406321	890608
17	थ	U+0925	THA	49234	196600	103027	240310
18	द	U+0926	DA	326974	616260	483036	501570
19	ध	U+0927	DHA	66748	257805	234150	186570
20	न	U+0928	NA	732500	1436484	893137	1575336
21	न्	U+0929	NNNA	12	0	2	15
22	प	U+092a	PA	495009	882660	647863	799293
23	फ	U+092b	PHA	72590	91216	43313	105543
24	ब	U+092c	BA	192114	390376	219139	365391
25	भ	U+092d	BHA	104808	275698	150777	314336
26	म	U+092e	MA	463270	1125913	762372	1067395
27	य	U+092f	YA	843198	1028144	1290958	769137
28	र	U+0930	RA	1019969	2568118	1607718	2021411
29	र्	U+0931	RRA	1950	0	24286	36
30	ल	U+0932	LA	828513	828206	951192	968434
31	ळ	U+0933	LLA	171446	0	134542	74
32	ळ	U+0934	LLLA	66	0	12	1
33	व	U+0935	VA	585202	825117	826984	569837
34	श	U+0936	SHA	237959	395921	301402	234032

Sr.	Consonant			Frequency			
No.	Character	Unicode	Devanagari Letter	Konkani	Hindi	Marathi	Nepali
35	ष	U+0937	SSA	36251	262928	185283	186098
36	स	U+0938	SA	598533	1393788	839395	1000627
37	ह	U+0939	HA	326585	861773	668410	600298

Table 3.3: Consonants and their frequency in Konkani, Hindi, Marathi and Nepali

3.5.2 Consonants

Consonants are characters that represent various consonant sounds. There are 37 consonant characters in Devanagari¹. Table 3.3 shows the consonants and their respective frequencies in four languages. It is observed that 32 consonants are common among these languages e.g. क [kə], ख [kʰə], ग [gə], घ [gʰə], च [tʃə], छ [tʃʰə], ज [dʒə], झ [dʒʰə], ञ [ɟə], ट [tə], ठ [tʰə], ड [də], ढ [dʰə], ण [ɳə], त [tə], थ [tʰə], द [də], ध [dʰə], न [nə], प [pə], फ [fə]/[pʰə], ब [bə], भ [bʰə], म [mə], य [jə], र [rə], ल [lə], व [və], श [ʃə], ष [ʃə], स [sə], ह [hə]. And five consonant ङ [ŋə], ण [ɳə], र [rə], ऌ [ɐ̃], ड [ɒ̃] does not appear in some language corpus. For specific examples in each language, refer to Table A.2 (refer Appendix A; Section A.2).

Occurrence of consonant ऌ and र Konkani and Marathi has more instances of ऌ and र. Nepali has rare instances, while Hindi does not have instances of same. E.g. र occurs in Konkani word गय्यो [gəɾjo] and Marathi word दुसऱ्या [dusɾja]. It is important to note that this य *ligature/syllable* can be formed using different character combinations as listed.

1. यः Consonant र (U+0931) with *halant* ् and consonant य [jə].
2. यः Consonant र (U+0930) with *halant* ् with Zero Width Joiner (ZWJ) and consonant य [jə].
3. यः Consonant र (U+0930) with *Nukta* ृ with *halant* ् and consonant य [jə].

These are the different ways the same syllable/*ligature* can be written. It is observed that there exist rare instances of *ligature* य formed with the characters shown in example 3.

Occurrence of consonant ङ ङ Nepali has more instances of ङ [ŋə] (e.g.: जनसङ्ख्या [dʒənsɒkʰja]). Marathi and Konkani have fewer instances, while Hindi does not have any instances. (refer Table A.2, section A.2 e.g. word वाङ्निश्चय [vaɳniʃtʃjə] in Marathi)

Occurrence of consonant ञ It is also observed that Nepali has rare instances of consonant ञ (for example refer Table A.2, section A.2). Marathi and Konkani also have rare instances of ञ. E.g. word घेऊन [gʰeu.n] is observed in Marathi. However, the word घेऊन has occurred only two times, it is a language-specific word, often spelled as घेऊन, with the same meaning. Consonant ञ is typing mistake in Marathi and Nepali. Hindi does not have any instances.

Occurrence of consonant ऌ Konkani has some instances of ऌ. Marathi has rare instances, while Hindi does not have any instances. In Nepali, there exists only one word for consonant ऌ (e.g. पाळि [paɭi]). consonant ऌ is typing error in Konkani and Marathi.

3.5.3 Dependent Vowels and various Devanagari signs

Dependent vowels are crucial characters that represent vowel sounds, and they are usually depicted as diacritic marks positioned around a consonant character. These dependent vowels indicate the precise pronunciation of vowels within a syllable¹³ that results in combining a consonant with a vowel sound.

3.5.3.1 Dependent Vowels

There exists a total of 20 distinct dependent vowels within Devanagari as shown in Table 3.4. It provides valuable information about these dependent vowels, including their respective frequencies in Konkani, Hindi, Marathi and Nepali. Table A.3 (refer Appendix A; Section A.3) provides an example of the same. These dependent vowels and signs on combining with Independent vowel or Consonant forms a syllable¹⁴. It is observed that out of 20 dependent vowels 11 are common in all four languages. And other dependent vowels occurrences are discusses here.

Occurrence of 2 dependent vowel ॄ (U+0944) and ो (U+094A): There are rare occurrences

¹⁴Independent vowel + Modifier:(vowel + dependent vowel or Devanagari sign (e.g. अं, ऊँ, आः, ऐं, औं, आ) or Consonant + Modifier: (Consonant + dependent vowel or Devanagari sign (e.g. का, कि, की))¹³

of ृ (U+0944) and ो (U+094A) in Marathi and Nepali, which are not seen in Konkani and Hindi. For example in Marathi मृत्यू [mr̥t̪ju] and Nepali सांस्कृतिक [sāskr̩tik]. Example for vowel ो in Marathi डोळे (gloss: eyes) and Nepali बसोबास (gloss: sit down). In Konkani, there is no instance of vowel ो in the Gold standard corpus used for this study.

Occurrence of dependent vowel ॅ: The dependent vowel ॅ has rare occurrence in Hindi specifically only six occurrences found in Hindi corpus. And all these occurrences are borrowed English words an example मॅनेजमेंट [mɛnɛdʒmɛ̃nt].

Occurrence of dependent vowel े: The dependent vowel े has three occurrences in Konkani, 102 in Marathi and 18 in Nepali and no occurrence in Hindi. For example in Marathi पट्टे ([pəʈt̪ɐ] gloss: strips) and वडे ([vəɖɐ] gloss: dish). An example in Konkani is सर्वनामांचें [sərvənāmāts̪ɛ̃]. An example in Nepali is खर्चे ([kʰət̪ɕɐ] gloss: spent) and मिनेट ([minɐt̪] gloss: minute). This dependent vowel े is typing error in Konkani and Marathi.

Dependent vowels not occurred in all languages: Five dependent vowels ँ (U+093A) and ऌ (U+093B), ड (U+094E), ऋ (U+094F) and ॠ (U+0955) are not commonly found in these four languages.

3.5.3.2 Devanagari Signs

There are six Devanagari Signs as shown in Table 3.5 namely i) *Candrabindu* ँ , ii) *Anusvara* ं , iii) *colon* ः , iv) *Nukta* ण, v) *avagraha* ॥, and vi) *inverted candrabindu* ॆ. Table 3.5 shows the frequency of these 6 Devanagari signs in Konkani, Hindi, Marathi and Nepali. It is observed that Devanagari sign/character ँ (U+0901), ं (U+0902), ः (U+0903) and ण (U+093C) has occurrences in all four languages.

Occurrence of sign ॆ: Devanagari sign ॆ (U+0900) is not occurring in any of four languages, and the vowel ॥ (U+093D) is exclusively present in Konkani and Nepali. Example for vowel ॥ in Konkani नासच [na:ʈs̪], वोसाः [vosɑ:], and in Nepali मदीयोऽयं [məp̪[di:jo:ɟ̃ə], सामवेदोऽस्मि। [saməvedo:smi] which language specific words. This sign is common in Sanskrit mainly used to increase the pronunciation length of the character.

Occurrence of sign ॥: Devanagari sign *avagraha* ॥ has occurrences only in Konkani and Nepali.

Sr. No.	Char-acter	Unicode	Vowel Sign ¹	Frequency			
				Konkani	Hindi	Marathi	Nepali
1	ा	U+093e	AA	2498524	2808069	3403896	2847444
2	ि	U+093f	I	440553	1556497	774496	1399254
3	ी	U+0940	II	761700	1237687	1083002	530126
4	ु	U+0941	U	228741	377078	330862	593040
5	ू	U+0942	UU	220886	227215	252669	217670
6	ृ	U+0943	VOCALIC R	10737	55744	35557	54344
7	ॠ	U+0944	VOCALIC RR	0	0	142	46
8	ँ	U+0945	CANDRA E	33564	6	12620	295
9	े	U+0946	SHORT E	3	0	102	18
10	ै	U+0947	E	978569	1958950	1419956	926419
11	ौ	U+0948	AI	15846	336985	31520	184979
12	ॉ	U+0949	CANDRA O	26350	46808	19015	331
13	ो	U+094a	SHORT O	0	0	28	26
14	ॊ	U+094b	O	538865	937614	451140	1023844
15	ो	U+094c	AU	5492	70173	17545	47366
16	ॅ	U+093a	OE	0	0	0	0
17	ॆ	U+093b	OOE	0	0	0	0
18	िॄ	U+094e	PRISHTHAMATRA E	0	0	0	0
19	ौॄ	U+094f	AW	0	0	0	0
20	ँॄ	U+0955	CANDRA LONG E	0	0	0	0

Table 3.4: Dependent Vowels Sign and their frequency in Konkani, Hindi, Marathi and Nepali

3.5.4 Additional Consonants and Vowels

Additional consonants and vowels appearing in Devanagari¹ are listed in Table 3.6. It is observed that these characters do not appear to be utilized much in Konkani, Hindi, Marathi and Nepali.

Occurrence of Additional Consonants क, ग, ज, ड, ढ, and ऋ: The consonant क has only one instance in the Marathi corpus used for this study. For example, अशफाक ([əʃəfak] gloss: pity), it's often spelled as अशफाक [əʃəfak] and occurs more frequently in the Marathi corpus. Similarly,

Devanagari Signs			Frequency			
Character	Unicode	Devanagari Sign	Konkani	Hindi	Marathi	Nepali
ँ	U+0901	CANDRABINDU	6191	9	4941	119074
ं	U+0902	ANUSVARA	1100133	1366490	687798	143382
ः	U+0903	VISARGA	740	2904	3491	8299
ॠ	U+093c	NUKTA	410	101383	634	506
ॡ	U+093d	AVAGRAHA	767	0	0	199
ं	U+0900	INVERTED CANDRABINDU	0	0	0	0

Table 3.5: Devanagari signs and their frequency in Konkani, Hindi, Marathi and Nepali

the consonant ण appears in the word गडकरीजीच्या ([gəḍəkəridzĩtsya] gloss: Gadkari's (noun)), and no other spelling besides this exists in the Marathi corpus used for the study. The consonant ज्ञ is found in the word हजार ([ɦəɖzar] gloss: thousand), which is also written as हजार [ɦəɖzar], and this variant is more common in the Marathi corpus used for this study. There are only a few words in Marathi that contain the consonant ड़, such as जोड़नीविणा ([dʒoḍɳivina] gloss: without connection) and ओउड़ [ouḍ], लड़ाऊँ [ḍaũ:] (gloss: Oud(place), fighter). Regarding the consonant ढ़ in Marathi, appears in words like सुदृढीकरण, आर्टिफिशियल, and फ़ appeared in words like फिलिपीन्स (gloss: Philippines) exist. In some cases, this word is written as फिलीपिन्स.

Occurrence of Additional Consonant यः The character यः (U+0959), which has occurrence in Konkani e.g. आयलें, आयली. In Konkani these words are often written as आयलें [ajəlē], आयली [ajəli]. Hence in Konkani it is typing error. There are few words in Marathi with consonant यः an example विद्यार्थ्यांनी (gloss: by the students), which is mostly written as विद्यार्थ्यांनी [vidjəɾəṭh̃jāni].

Occurrence of Additional Consonant ऋः The additional consonant ऋ (U+0960) used in Marathi for example in word ऋतूमध्ये ([ɾuṭuməḍh̃je] gloss: in season). In Nepali it has occurrences in words like ऋषि-श्रापवश, ऋग्वेदीय, ऋग्वेद, कृपाचाऋ, काऋ

Occurrence of Additional Consonants through its canonical equivalence form (Consonant + Nukta e.g. ढ (U+0922)+ ॠ (U+093C)): Table 3.5 showed the large number of instances of Nukta in Hindi, which exhibited further analysis. This revealed the existence of character ढ़ which is combination of ढ (U+0922)+ ॠ (U+093C), which is canonical equivalence to character ढ़ (U+095D). Other examples of the same in the Hindi corpus are ड़, ड़े, ज़ा. Thus, although

additional characters do not exist in Hindi, its canonical equivalence syllable occurrences have been seen. For example word टंगड़ी [t̪əŋgəḍi] ङ has Canonical Equivalence. It is important to note that many of rarely occurring characters in Marathi and Konkani are typing errors.

Additional consonant and vowel sign			Frequency			
Character	Unicode	Devanagari Letter	Konkani	Hindi	Marathi	Nepali
Additional Consonant						
क	U+0958	QA	0	0	1	0
ख	U+0959	KHHA	0	0	0	0
ग	U+095a	GHHA	0	0	1	0
ज	U+095b	ZA	0	0	1	0
ङ	U+095c	DDDHA	0	0	3	0
ढ़	U+095d	RHA	0	0	1	0
फ़	U+095e	FA	0	0	6	0
य़	U+095f	YYA	232	0	3	0
Additional vowels for Sanskrit						
ऋ	U+0960	VOCALIC RR	0	0	24	112
ॠ	U+0961	VOCALIC LL	0	0	0	1
ॡ	U+0962	VOCALIC L	0	0	0	0
ॢ	U+0963	VOCALIC LL	0	0	0	0

Table 3.6: Additional Consonants & Vowels for Sanskrit and their frequency in Konkani, Hindi, Marathi and Nepali.

3.5.5 Punctuation marks and others

Table 3.7 shows that Danda | (U+0964) has been widely used in Hindi and Nepali and rare occurrence seen in Konkani and Marathi. Additionally the Vedic tone , Accent marks and additional Devanagari characters are not used in Konkani, Hindi, Marathi and Nepali. The ॐ character is used by all these languages, at present there is no instances of this character in the corpus used for this study.

Sr. No.	Generic punctuation			Frequency			
	Character	Unicode	Devanagari Letter	Konkani	Hindi	Marathi	Nepali
1		U+0964	DANDA	567	286633	175	347857
2		U+0965	DOUBLE DANDA	0	0	0	393
3	ॐ	U+0950	OM	15	0	0	380

Table 3.7: Generic punctuation and their frequency in Konkani, Hindi, Marathi and Nepali

3.5.6 Viram/Halant

The *halant* is a diacritic mark used in several Indian scripts, including Devanagari, to suppress the inherent vowel sound of a consonant. In the context of the Devanagari script, the halant is also known as the *virama*. It looks like a horizontal line, and when placed after a consonant character, it indicates that the consonant should be pronounced without its inherent vowel sound. The Devanagari consonant क (*ka*) with the inherent vowel *a* sounds like *ka*. When you add the halant to it, you get क् (*k*), which is pronounced as *k* without the *a* sound. The *halant* is a crucial diacritic mark in Devanagari and other Indian scripts, as it plays a significant role in representing the precise pronunciation of words and forming complex consonant clusters $\overline{\text{कक}}$ [k̐k̐], ल्ह [lvhə], ड्ट [d̐t̐ə] in Indian languages. Table 3.8 shows the frequency of *viram/halant* appearing in each language. This indicates that many cluster of consonant (*ligatures*) appear in each of the languages. The different form of *ligatures* it can form are conjunct consonant¹³ (e.g. $\overline{\text{कक}}$), Multi-Consonant Conjuncts³ (e.g. ल्ह), Preventing Conjunct Forms³ (e.g. ड्ट), Dead Consonants³ (e.g. क्). These *ligatures* can combine with dependent vowels and Devanagari signs forming syllable (*ligature-vowels-signs-diacritic*) (e.g. ह्या [hja], वह [vhə], लीं [lī̃])¹³.

Devanagari Sign			Frequency			
Character	Unicode	Devanagari Sign	Konkani	Hindi	Marathi	Nepali
्	U+094d	VIRAMA	1128012	2123026	2112285	2328783

Table 3.8: Viram sign and its frequency in Konkani, Hindi, Marathi and Nepali

3.6 Devanagari syllables (*ligature-vowels-signs-diacritic*) and their frequency

These *ligatures* can combine with dependent vowels and Devanagari sign that forms syllable¹³. Some of the examples, along with their frequency in Konkani, Hindi, Marathi, and Nepali, are seen in Table 3.9 (For more examples, refer to Table A.4 (see Appendix A; Section A.4)). It is observed that differences do exist in each of the languages. However these syllable consist of consonant conjunct ल [llə] appearing in form of लीं [lī], ल as well as ल्या [lja] and in more consonant cluster form. Thus it was important to study the frequency of *ligatures* (only consonant cluster).

Sr. No.	Syllable (<i>ligature-vowels-signs-diacritic</i>)	Frequency			
		Konkani	Hindi	Marathi	Nepali
1	व्ह	10718	4	3505	0
2	र्सा	5601	0	0	0
3	र्डा	199	0	14	0
4	लीं	3232	1	4	0
5	श्च	119	214	846	683
6	भ्र	1511	833	851	1463
7	द्ध	1589	232	7125	1125
8	ल्यां	884	0	109	0

Table 3.9: Example of syllables (*ligatures-vowels-signs-diacritic*) and their frequency in Konkani, Hindi, Marathi and Nepali

3.7 Devanagari *ligatures* and their frequency

Table 3.10 shows the frequency of some *ligatures* occurring in Konkani, Hindi, Marathi, and Nepali. For more examples, refer to Table A.5 (refer Appendix A; Section A.5). It is observed that there exist differences between each of the languages.

Sr. No.	Ligatures	Frequency			
		Konkani	Hindi	Marathi	Nepali
1	ळ्य	14786	0	6238	1
2	ळळ	5132	0	6	3
3	ळळ्य	868	0	0	0
4	श्श	254	0	851	9
5	ण्ट्य	187	0	0	0
6	ट्ट्य	164	0	206	45
7	ज्य	30	0	7	1
8	द्र्य	121	0	69	0

Table 3.10: Some example of *ligatures* whose occurrence is zero in one of the languages

3.8 Observation and Discussion

Binary Jaccard similarity score is used to measure the similarities and differences of Characters, syllables and *ligatures* among the Konkani, Hindi, Marathi and Nepali. A Binary Jaccard similarity score is used to calculate the similarities.

Evaluation metric The following is equation to calculate binary Jaccard score:

$$\text{Jaccard}(U, V) = \frac{|U \cap V|}{|U \cup V|},$$

Where U is a binary vector where each element corresponds to a Devanagari character. The value is set to 1 if the character exists in the first language, and 0 if it does not. V is a binary vector similar to U, representing the second language. Each element in V is 1 if the corresponding Devanagari character exists in the second language, and 0 if it does not. The Jaccard Similarity was computed for Independent vowels, Consonants, Dependent vowels, Devanagari signs, syllables (*ligature-vowels-signs-diacritic*) and *ligatures* among the languages: Konkani, Hindi, Marathi, and Nepali, as detailed in Table 3.11. The results indicate that the highest binary similarity score for Independent vowels between Konkani and Nepali is 0.93, and between Marathi and Nepali is 0.94. Further, the binary Jaccard similarity score for consonants among Konkani, Marathi, and Nepali is 1.0 (Table 3.11).

Further binary similarity scores for syllables (*ligature-vowels-signs-diacritic*) range from 0.27

#	Devanagari Character set, syllables and <i>ligatures</i>	Konkani			Hindi		Marathi
		Hindi	Marathi	Nepali	Marathi	Nepali	Nepali
A	Independent Vowels	0.85	0.87	0.93	0.75	0.80	0.94
B	Consonants	0.86	1.0	1.0	0.86	0.86	1.0
C	Dependent Vowels	0.92	0.86	0.86	0.80	0.80	1.0
	Devanagari signs	0.80	0.80	1.0	1.0	0.80	0.80
D	Syllables (<i>ligatures</i> and <i>ligature-vowels-signs-diacritic</i>)	0.34	0.40	0.27	0.39	0.30	0.29
E	<i>Ligatures</i>	0.34	0.36	0.27	0.37	0.25	0.24
Overall (includes A,B,C and D)		0.35	0.41	0.28	0.40	0.31	0.30

Table 3.11: Binary Jaccard Similarity score for Devanagari characters, syllables and *ligatures* between each of the languages: Konkani, Hindi, Marathi and Nepali

to 0.40. The highest binary similarity score is observed between Konkani and Marathi, which is 0.40. However, this shows that differences exist between these languages. Additionally, the binary similarity score is calculated for *ligatures*, which shows a score close to that of syllables (*ligature-vowels-signs-diacritic*). This indicated that the *ligatures* vary for among these languages. The overall binary Jaccard Similarity among Konkani, Hindi, Marathi, and Nepali is presented in Table 3.11, which includes the categories A, B, C, and D (refer Table 3.11). It is observed that Konkani has a higher similarity with Marathi, attaining a binary Jaccard score of 0.41. Hindi is more similar to Marathi, with a binary Jaccard score of 0.40. Nepali is more similar to Hindi, with a score of 0.31. However, these overall scores indicate that differences exist between these languages.

3.9 Conclusion

This study highlights the similarities and differences between Konkani, Hindi, Marathi, and Nepali. The highest binary Jaccard similarity score of 0.41 for overall characters set and syllables (*ligatures* with dependent vowels and sign) in Konkani and Marathi. Hindi resembles Marathi more closely with a binary Jaccard score of 0.40. Moreover, Marathi showcases a higher similarity to Konkani, with a score of 0.41, while Nepali displays a closer resemblance to Hindi,

with a score of 0.31. However, it is crucial to note that dissimilarities, as suggested by the binary Jaccard similarity score, signify that each language possesses unique syllables and *ligatures*.

Chapter 4

Datasets Created for OCR Performance study

4.1 Background and Motivation

Understanding the character usage varies for each of the language and its appearance differs for different fonts. Devanagari OCRs are available specific to language (refer Chapter 2; Section 2.2). The available Hindi OCR test dataset ([Mathew et al., 2016](#)) is in single fonts. To test the performance of OCR for different fonts, it is essential to create the same dataset in different fonts. Further, no dataset was available for Konkani, Marathi, and Nepali for OCR and OCR post-processing. Thus, creating a datasets to conduct this research was essential.

This chapter discusses the dataset created for OCR and OCR post-processing for Hindi, Konkani, Marathi, and Nepali. It is organized as follows; Section 4.2 discusses the proposed manual procedure to create the OCR dataset for different fonts. Section 4.3 has six parts; Section 4.3.1 describes the proposed *text2img-doc* procedure to create an OCR dataset for any given text specifying the fonts and font size. Section 4.3.2 describes *post-OCR-align* procedure to create OCR post-processing dataset. Section 4.3.3 details the OCR post-processing dataset created for Hindi. Section 4.3.4, details the characteristics of data and dataset created to study the OCR performance for Konkani. Section 4.3.5 provides details of dataset created for Konkani having ten fonts. Section 4.3.6 discusses two datasets; first, it provides details of the dataset created to evaluate OCR for four languages. The second dataset was created to train and test OCR post-processing approaches for four languages: Konkani, Hindi, Marathi, and Nepali, as discussed. Finally section 4.4 concludes the chapter.

- a) (मंच पर एक कमरा जिसमें थोड़ा सा फर्नीचर
b) (मंच पर एक कमरा जिसमें थोड़ा सा फर्नीचर

Figure 4.1: A sample text in a different font style a) *Aparajita* and b) *Nakula*.

4.2 Using Manual Procedure

A manual procedure has been proposed and used to create OCR dataset for *Aparajita* and *Nakula* font. Figure 4.1 shows the text sample in *Aparajita* and *Nakula* fonts. The *Aparajita* and *Nakula* fonts were selected because they have distinct properties, and text with similar fonts appears in newspapers and books (see Chapter 1; Figure 1.7). Here is the procedure to create OCR dataset:

1. N number of text files and name of the font *font-name* are taken as input.
2. Each text is placed in *Word* document file *word-doc-file*. Verifies whether each text file fits on each page.
3. Further Specific *font-name* font is applied for the *Word* document file *word-doc-file*.
4. Further, *word-doc-file* was printed on 75 GSM A4 and scanned at 300 dpi using the Konica Minolta Scanner to create images. *Note: This 75 GSM A4 paper is selected because it is commonly used for printing, and 300 dpi is chosen since OCR accuracy is typically demonstrated on images with 300 dpi resolution.*
5. The output is N number of image files in *png* format

4.2.1 Hindi datasets for *Aparajita* and *Nakula* fonts.

4.2.1.1 Characteristics of data

Hindi OCR test dataset (Mathew et al., 2016) consists of 100 image files and 100 corresponding text files in single font. The characteristics of this data (Mathew et al., 2016) is outlined in Table 4.1. It shows the distribution of words based on the type of character present in the word. There are a total of eight types of categories of words in the dataset. Basically, the complexity of characters affect the accuracy of OCR. Basically, the complexity of characters affect the accuracy

Category	No of words in Hindi data
<i>Without modifiers</i>	9930
<i>Matras</i>	7849
<i>Eekar</i>	4723
<i>Ookars</i>	1340
<i>Conjunct consonant</i>	1083
<i>Anusvara</i>	248
<i>Candra</i>	298
<i>Words with multiple modifiers</i>	5868
Total	31339

Table 4.1: Characteristics of available Hindi data to test OCR

of OCR. These categories were created to ensure the dataset comprehensively includes all types of Devanagari characters used in the languages selected for this study. The description to each of the categorizes is as follows:

- (i) *Without modifier*: Words with Devanagari numerals or vowels or consonants with no modifiers. e.g. मन ([mən]), हलचल ([həltʃl]), 2023.
- (ii) *Matra*: Words with *Matra*¹ e.g. वैर ([vairə]).
- (iii) *Eekar*: Words with modifier *Eekar*² e.g. सीप ([sip]).
- (iv) *Ookar*: Words with modifier *Ookar*³ e.g. अनुभव ([ənubʰəv]).
- (v) *Conjunct consonant*: words, which have *Conjunct consonant*⁴ e.g. पल्लव ([pəlləv]).
- (vi) *Anusvara*: Words having *Anusvara*⁵ e.g. पंप ([pəp]).
- (vii) *Candra*: Words with *Candra*⁶ e.g. चाँद ([tʃāṁd]).

¹Matra is a Dependent vowel sign. े ([e]), ै ([ai]), ो ([o]), and ौ ([au]) in Devanagari (Consortium, 2022)

²Eekar is a Dependent vowel sign. ि ([i]) and ी ([e:]) in Devanagari (Consortium, 2022)

³Ookar is a Dependent vowel sign. ु ([u]) and ू ([u:]) in Devanagari (Consortium, 2022)

⁴Conjunct consonant have more than two consonant letters (Shapiro, 1989)

⁵Devanagari sign ँ (nasal sound) (Consortium, 2022)

⁶Devanagari sign ऌ and letter "Candra A".(Consortium, 2022)

- (viii) *Words with multiple modifiers*: Words which have more than one modifier *Ookar* or *Anusvara* or *Eekar* and does not belong to any of the other categories e.g. मणिका ([mən̪ika]), नितारा (niṭara).

The following two datasets were created:

4.2.1.2 *BM_Aparajita* and *BM_Nakula*

The standard Hindi OCR test dataset having 100 original text files (Mathew et al., 2016) (Table 4.1 shows its characteristics i.e. number of words in each category) and *Aparajita* font were input to the procedure described in section 4.2, to create OCR dataset having text in *Aparajita* font. This dataset was named as *BM_Aparajita*. Same procedure is repeated for *Nakula* font to create OCR dataset in *Nakula* named as *BM_Nakula* dataset.

4.3 Using *text2img-doc* and *post-OCR-align* procedure

Two automated procedure are proposed; (a) *text2img-doc* procedure to create an OCR dataset (b) *post-OCR-align* procedure to create dataset for OCR post-processing.

4.3.1 Procedure *text2img-doc* for OCR.

The procedure to convert text to image documents is as follows, named as *text2img-doc*. Here, the inputs are text files, the font name, and the font size. The output is image files and corresponding text files.

1. The text in multiple text files was merged into a single file called *merged_text* file.
2. This *merged_text* file was split into N_text files, with each file containing 22 lines, so that text would fit on an A4 size paper.
3. Further, these N_text files were converted to create N_Img files in ".png" format with $\langle name\ of\ the\ font \rangle$ (e.g. *Nakula* font) with a $\langle font-size \rangle$ and 150 DPI for "X" and "Y"

resolution ⁷ ⁸.

4.3.2 Procedure *post-OCR-align* for OCR post processing

The following is the procedure to create a post-processing dataset named *post-OCR-align*. Here the input is ground truth N_text and N_Img files and output is $N_Aligned_Text$ files.

1. The ground truth text files N_Img files were given as input for Tesseract OCR (Hindi). The output of the OCR process was a collection of corresponding OCR-recognized text files, which is referred to as N_OCR files.
2. In order to assign a label (0 for correct, 1 for incorrect) to each OCR recognized word, the original N_text and N_OCR files were aligned using Recursive Text Alignment system (RETAS) (Gupte et al., 2021).
3. Output is $N_Aligned_Text$ files. The aligned text files were formatted as per the structure provided in a dataset of the ICDAR Competition on Post-OCR Text Correction (Rigaud et al., 2019). This structure specifies that the file should contain three parts (a) [OCR_toInput], which is raw OCR'd text to be de-noised, (b) [OCR_aligned] which has Aligned OCR'd text and (c) [GS_aligned], which has aligned Original text. The GS_aligned and OCR_aligned text have the same length. The missing characters are defined by "@" sign.

4.3.3 Hindi datasets for OCR post-processing.

4.3.3.1 Characteristics of data

There is no Hindi test dataset available to test error correction approaches. Thus two dataset are created one with standard Hindi data (Mathew et al., 2016) and other with part of Hindi data (Shivam, 2022) (refer Table 4.2 for properties of data).

⁷<https://linux.die.net/man/1/paps>

⁸<https://manpages.debian.org/testing/poppler-utils/poppler-utils.pdftoppm.1.en.html>

Word category	Frequency of words in test dataset	
	<i>Hin_test_dataset</i>	<i>Hindi-IIITH-dataset</i>
<i>Without modifiers</i>	5274	9930
<i>Matras</i>	7140	7849
<i>Eekar</i>	4872	4723
<i>Ookars</i>	792	1340
<i>Conjunct consonant</i>	1869	1083
<i>Anusvara</i>	571	248
<i>Candra</i>	65	298
<i>Words with multiple modifiers</i>	9301	5868
Total	29884	31339

Table 4.2: Characteristics of data used for *Hin_test_dataset* and *Hindi-IIITH-dataset* to test error correction approach

4.3.3.2 *Hindi-IIITH-dataset* and *Hin_test_dataset*

Hindi text data from standard OCR test dataset ([Mathew et al., 2016](#)), *Nakula* font and 12pt font size were input to *text2img-doc* procedure (refer section 4.3.1) thus it created *N_Img* and *N_text* files. Further these files are given input to procedure *post-OCR-align* and the output is *N_Aligned_Text* files. Thus, the *Hindi-IIITH-dataset* was created to test the error correction approach.

The part of Hindi data ([Shivam, 2022](#)) was taken and verified the properties of data (number of words category-wise) with the text of standard OCR dataset ([Mathew et al., 2016](#)), as shown in Table 4.2. The same procedure was repeated for this text, creating *Hin_test_dataset*.

4.3.4 Konkani datasets for OCR and OCR post-processing

There is no OCR test dataset and OCR post-processing test dataset available for the Konkani. Thus, to create the Konkani test dataset, the gold standard Konkani raw corpus from Linguistic Data Consortium for Indian Languages (LDC-IL) ([Ramamoorthy et al., 2019](#)) was used to create dataset. It consists of 3619017 words.

4.3.4.1 Characteristics of data

Categories	Total words		Unique words	
	Frequency	Percentage	Frequency	Percentage
<i>Simple syllable (akshara)</i>	614514	16.98%	34670	8.62%
<i>Matra</i>	493604	13.64%	38892	9.67%
<i>Eekar</i>	407980	11.27%	28346	7.05%
<i>Ookar</i>	152945	4.23%	12000	2.98%
<i>Conjunct consonant</i>	291381	8.05%	21100	5.25%
<i>Anusvara</i>	190984	5.28%	12567	3.13%
<i>Candra</i>	14339	0.40%	1884	0.47%
<i>Words with multiple modifiers</i>	1453270	40.16%	252580	62.82%
Total	3619017	100%	402039	100%

Table 4.3: Properties of gold standard Konkani raw text corpus from LDC-IL

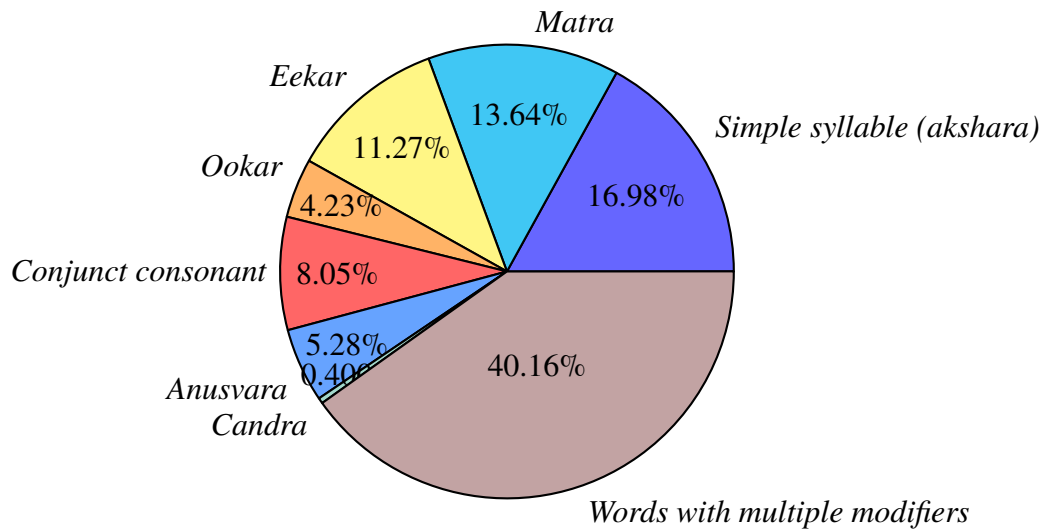


Figure 4.2: Category-wise distribution of total words (3619017 words) in gold standard Konkani LDC-IL corpus

The properties of LDC-IL Data (3619017 words) is shown in Figure 4.2. The words were classified into eight categories as mention in section 4.2.1.1. The following are the examples in categories of the words in the corpus used to create the dataset. These categories are not standard but were created specifically for this study to capture all types of characters in the selected

Sr	Simple syllable (akshara)		Matra		Eekar		Ookar	
No	Word	Freq.	Word	Freq.	Word	Freq.	Word	Freq.
1	ना (no)	20626	तो (he)	15875	आनी (and)	65526	पूण (but)	16906
2	आसा (is there)	20326	ते (they)	14743	खातीर (because)	13165	करून (finished)	9847
3	एक (one)	18078	हे (this)	13853	ही (this)	10036	लागून (touched)	6764
4	ताका (him)	13575	हो (this)	10230	ती (she)	9723	खूब (many)	4535
5	जाय (need)	9819	कडेन (near)	8630	आमी (we)	7576	मुखार (front)	4478
6	तर (then)	6284	दोन (two)	5616	तरी (but)	6435	सुरु (start)	2926
7	वा (or)	5830	घेवन (taken)	5491	अशी (like this)	4802	पुराय (full)	2506
8	चड (more)	5460	वेळार (time)	4556	मागीर (later)	4023	तुका (you)	2478
9	एका (once)	5440	असो (like this)	4320	दीस (day)	3621	आपूण (self)	2453
10	भायर (out)	5147	मेरेन (with)	3936	तिका (her)	3196	वचून (go)	2088

Sr	Conjunct consonant		Anusvara		Candra		Words with multiple modifiers	
No	Word	Freq.	Word	Freq.	Word	Freq.	Word	Freq.
1	ह्या (this)	21474	हांव (me)	9628	मॅच (match)	2490	अशें (like this)	18646
2	जाल्यार (can)	12378	कांय (what)	8236	डॉ (Dr.)	1473	हें (this)	13745
3	म्हाका (myself)	9529	आतां (now)	6643	बॉल (ball)	696	तें (that)	12359
4	त्या (those)	8889	उपरांत (afterwards)	6419	बॉलर (bowler)	644	कितें (what)	7466
5	आपल्या (your)	5792	तांकां (them)	5936	मॅन (man)	232	म्हूण (say)	6248
6	न्हय (river)	4806	थंय (there)	4173	कॅच (catch)	199	तांणी (those)	6130
7	म्हणल्यार (said)	3619	आमकां (us)	3893	जॉन (Jhon)	182	तेन्ना (at that time)	6042
8	ताच्या (his)	3418	हांगा (here)	3048	बॉ (bow)	173	ताणें (he)	5621
9	म्हज्या (my)	3158	जावंक (happen)	2717	डॉन (don)	159	लेगीत (near)	4960
10	आमच्या (our)	2972	पंगड (group)	2400	टॉस (toss)	131	सांगलें (said)	4893

Table 4.4: Category wise top 10 frequent word appearing in *Konkani_OCR_large_test_dataset*

languages, as OCR accuracy is influenced by the complexity of the Devanagari script i.e the way it is written.

- (i) *Simple syllable (akshara)*: This category consists of words without any dependent vowels, Devanagari signs and halant. Like words with Devanagari numerals or vowels or consonants without अ ([ə]) or आ ([a]) modifiers. e.g. नाल ([na:l] 'coconut'), हात ([ha:t] 'hand'), 2023.
- (ii) *Matra*: e.g. वेळार ([vɛʃar] 'on that time'), मोर ([mor]/[mɔr] 'peacock/peacocks').
- (iii) *Eekar*: e.g. जीवित ([dzi:vɪt] 'alive'), गजाली ([gədzali] 'casual conversation').

Sr.No	Bigram of words	Frequency
1	ताका लागून	1828
2	आसा अशें	1379
3	अशें तांणी	1029
4	आनी हेर	1019
5	ना अशें	1016
6	ह्या वेळार	997
7	ना जाल्यार	964
8	ते खातीर	948
9	गरज आसा	891
10	एक दीस	766

Table 4.5: Top 10 bigrams of the words and its frequency from Konkani corpus LD-CIL (Ramamoorthy et al., 2019)

(iv) *Ookar*: e.g. गुलाब ([gula:b] 'rose'), फूल ([fu:l] 'flower').

(v) *Conjunct consonant*: e.g. ह्या ([hja:] 'this'), म्हाका ([mha:ka] 'for me').

(vi) *Anusvara*: Words having only Anusvara⁹ e.g. हांव ([hãv] 'me'), उपरांत ([upərãt] 'after-words').

(vii) *Candra*: Words with only *candra*¹⁰ modifier. This category has the lowest percentage of words in the corpus. This modifier is not used for native Konkani words but mainly in words borrowed from foreign languages such as English. e.g. बॉल ([bəl] 'ball'), बॉलर ([bələr] 'bowler').

(viii) *Words with multiple modifiers*: e.g. स्वतेज ([svəteɟz] 'with self brightness').

Table 4.3 shows the frequency and percentage of total and unique words in the entire corpus. Figure 4.2 presents category-wise distribution of words across different categories. Table 4.4 displays category-wise category's ten most frequent words. Table 4.5 shows the top 10 bigrams of words and its frequency in the Konkani corpus from LD-CIL (Ramamoorthy et al., 2019).

⁹Devanagari sign ँ (nasal sound) (Consortium, 2022)

¹⁰Devanagari sign ँ and letter "candra" (Consortium, 2022)

Sr.No	Bigram of syllables	Frequency
1	क र	41679
2	आ स	25565
3	आ प	23817
4	आ य	22739
5	व पा	22536
6	व न	20599
7	म्ह ण	19895
8	र ता	18363
9	प य	17042
10	र पा	16644

Table 4.6: Top 10 bigrams of the syllables and its frequency from Konkani corpus LD-CIL ([Ramamoorthy et al., 2019](#))

And top 10 bigrams of syllables along with its frequency in Konkani corpus from LD-CIL ([Ramamoorthy et al., 2019](#)) are shown in Table 4.5. The following are the three datasets created for the Konkani language.

4.3.4.2 Konkani_OCR_large_test_dataset

The LDC-IL text having 3619017 words, *Nakula* font, and 12pt font size were given as input to *text2img-doc* procedure and created image files and corresponding text files. This was named as *Konkani_OCR_large_test_dataset*.

4.3.4.3 Konkani_OCR_subset_test_dataset

The subset of LDC-IL text having 27067 words and whose properties were verified with data standard Hindi OCR test dataset ([Mathew et al., 2016](#)). This text, *Nakula* font and 12pt font size were input to *text2img-doc* procedure and output generated OCR test dataset for Konkani (named as *Konkani_OCR_subset_test_dataset*).

4.3.4.4 Konkani_EC_test_dataset

The text and image files from *Konkani_OCR_subset_test_dataset* was input to procedure *post-OCR-align* and output was *N_Aligned_Text* files, thus creating test dataset to evaluated OCR post-processing approaches, which is named as *Konkani_EC_test_dataset*.

4.3.5 Konkani dataset for Multi-font to test OCR (*Konkani_test_dataset*)

Sr. No.	Font name	Sample text
1	<i>Amiko</i>	हांवें एका पुस्तकांत वाचिल्लें
2	<i>Lohit Devanagari</i>	हांवें एका पुस्तकांत वाचिल्लें
3	<i>Biryani</i>	हांवें एका पुस्तकांत वाचिल्लें
4	<i>Nakula</i>	हांवें एका पुस्तकांत वाचिल्लें
5	<i>Cambay</i>	हांवें एका पुस्तकांत वाचिल्लें
6	<i>Eczar</i>	हांवें एका पुस्तकांत वाचिल्लें
7	<i>Gargi</i>	हांवें एका पुस्तकांत वाचिल्लें
8	<i>Asar</i>	हांवें एका पुस्तकांत वाचिल्लें
9	<i>Baloo</i>	हांवें एका पुस्तकांत वाचिल्लें
10	<i>Aparajita</i>	हांवें एका पुस्तकांत वाचिल्लें

Table 4.7: Konkani text in ten fonts

There is no multi-font to test OCR dataset for Konkani. Hence, the Konkani OCR test dataset consisting of 10 fonts: *Amiko*, *Lohit Devanagari*, *Biryani*, *Nakula*, *Cambay*, *Eczar*, *Gargi*, *Asar*, *Baloo*, and *Aparajita*, was created. The following procedure was used:

- The text files from *Konkani_OCR_subset_test_dataset* , *Ambiko* font and 12pt font size was given input to *text2img-doc* procedure creating images in *Ambiko* fonts.
- Above step was repeated for the other nine fonts, namely *Lohit Devanagari*, *Biryani*, *Nakula*, *Cambay*, *Eczar*, *Gargi*, *Asar*, *Baloo*, and *Aparajita*.

The dataset created was named as *Konkani_test_dataset*. Table 4.7 shows the sample Konkani text in 10 fonts.

4.3.6 Indic datasets for OCR and OCR post-processing

4.3.6.1 Indic-OCR-test-dataset

In order to test the OCR approach for more than one language written in Devanagari script, there was a need to create a dataset consisting of four languages, Konkani, Hindi, Marathi, and Nepali, with a general text corpus. Thus the general corpus was taken from Indian Languages Corpora Initiative (ILCI) corpus. The properties of data and source for each language are mentioned in Table 4.8. The dataset was named as *Indic-OCR-test-dataset*.

Word category	Number of words in language data			
	Konkani	Hindi	Marathi	Nepali
<i>Without modifiers</i>	5128	9930	5357	4831
<i>Matras</i>	4486	7849	4974	5260
<i>Eekar</i>	3753	4723	3548	3926
<i>Ookars</i>	1511	1340	1037	1009
<i>Conjunct consonant</i>	2454	1083	2840	2454
<i>Anusvara</i>	2108	248	557	22
<i>Candra</i>	32	298	21	408
<i>Words_with_Multiple_Modifiers</i>	13019	5868	12345	12441
Total	32491	31339	30679	30351

Table 4.8: Characteristics of data used to create *Indic-OCR-test-dataset*

Konkani text ie. *Konkani Monolingual Text Corpus ILCI-II* (TDIL, 2019b), *Nakula* font and 12pt font-size were input to *text2img-doc* procedure, thus creating OCR dataset for Konkani named as *Konkani-Indic-OCR-test-dataset*. The same was repeated for Marathi and Nepali where *Marathi Text Corpus ILCI-II* (TDIL, 2019a) and *Nepali Text Corpus TDIL* (TDIL, 2019c) was the input along with *Nakula* font and 12pt font size, thus creating OCR test dataset for Marathi and Nepali named as *Marathi-Indic-OCR-test-dataset* and *Nepali-Indic-OCR-test-dataset* respectively.

To create the Hindi dataset, the 100 text files from IIITH dataset (Mathew et al., 2016) were used, and the above procedure was repeated, which created the *Hindi-Indic-OCR-test-dataset*

dataset consisting of 100 text files and 100 image files. The intention behind creating the Hindi test dataset was to have a uniform procedure and standard text to evaluate OCR accuracy for each of the other languages. Thus *Indic-OCR-test-dataset* consists of four OCR datasets for each language.

4.3.6.2 Indic-post-OCR-dataset

There is no dataset available to create an error detection model supporting four languages Konkani, Hindi, Marathi and Nepali. Hence to create a dataset for all four languages, data for each language was gathered from different sources, as mentioned in Table 4.9. Further, the procedure *text2img-doc* and *post-OCR-align* was used to create *N_Aligned_Text* files. This dataset was named as *Indic-post-OCR-dataset*. This was used as train and test dataset for the error detection approach. Table 4.10 shows a sample text from one of the files in the *Indic-post-OCR-dataset*.

Language	Number of words	Source of data
Hindi	248901	<i>Word Frequency for Hindi</i> IIT Bombay (cfile.iitb, 2023) and <i>IIIT-H</i> (Mathew et al., 2016)
Konkani	178366	<i>Konkani Monolingual Text Corpus ILCI-II</i> (TDIL, 2019b) and part of Konkani LDC-IL (Ramamoorthy et al., 2019)
Marathi	265269	Marathi General Corpus (TDIL, 2019a) and English Marathi corpus (Singh, 2020)
Nepali	296908	<i>Nepali General Text Corpus TDIL</i> (TDIL, 2019c) and Nepali News data (Shahi and Pant, 2018)

Table 4.9: Data used for creating *Indic-post-OCR-dataset*

[OCR_toInput] रातचो घरा येवन केन्ना केन्ना उपाशीच न्हिदतालों.
[OCR_aligned] रातचो घरा येवन केन्ना केन्ना उपाशी@ च न्हिदतालों.
[GS_aligned] रातचो घरा येवन केन्ना केन्ना उपाशीच न्हिदतालों.

Table 4.10: Sample text in one of the file in the *Indic-post-OCR-dataset*

4.4 Conclusion

In this chapter three procedure are proposed to create datasets to study OCR performance. These procedures are used to create dataset for Hindi, Konkani, Marathi and Nepali language to study the performance of OCR and post processing approaches. This work lead to creation of datasets *BM_Aparajita*, *BM_Nakula*, *Hindi_test_dataset* and *Hindi-IIITH-dataset* for Hindi to study OCR performance. Additionally created dataset for Konkani *Konkani_OCR_large_test_dataset*, *Konkani_OCR_subset_test_dataset*, *Konkani_EC_dataset* and *Konkani_test_dataset* (supporting ten fonts) to study OCR performance. Moreover, created dataset supporting four Konkani, Hindi, Marathi and Nepali languages; *Indic-OCR-test-dataset*, *Indic-post-OCR-dataset*. Moreover multi-font datasets are created for Hindi and Konkani. These procedure can be utilized to create dataset for any languages.

Chapter 5

Algorithms Proposed and Implemented for OCR Error Correction

5.1 Background and Motivation

Currently these are OCR options are available for Hindi; Tesseract OCR, Indsenz OCR, e-Aksharayan and Multilingual OCR (showed 91.60% word accuracy on Hindi dataset for single font.) (refer Chapter 2; Section 2.2). OCR error correction approaches have been explored for Indian languages to improve the OCR accuracy. These include morphological analyzer(for Bangla 82.44%), shape-based method (for Gurmukhi CA 97.34%), Locality Sensitive Hashing (LSH) (WA 79.12%), Graph-based sub-character level (Malayalam WA 95%), statistical language models (SLM) and clustering words in documents. Deep learning techniques such as LSTM for Hindi, Malayalam, Sanskrit, and Kannada, sub-Word embedding (Hindi, Sanskrit, Kannada, and Malayalam), and Word Embedding and Levenshtein Distance for Hindi has also been reported (refer Chapter 2; Section 2.3.2). Hence there is scope for improving Devanagari OCR accuracy.

OCR for Hindi is available (refer Chapter 2; Section 2.2) and tested for the single font ([Mathew et al., 2016](#)). Presently, through this work, OCR Hindi datasets for the multi-font are created (i.e. *BM_Aparajita* and *BM_Nakula*) (refer Chapter 4; Section 4.2.1.2). There is no study showing the performance of OCR for multi-font datasets. Moreover, no Multi-font OCR has been reported for Hindi. Additionally, through this work, datasets for Konkani are created (refer Chapter 4; Section 4.3.4). Currently, an OCR solution specific to Konkani is not available, whereas they are available for Hindi and Nepali. Additionally, various error correction approaches have been

explored for Hindi. However, no error correction work has been reported handling errors for different fonts. Furthermore, these existing error correction approaches do not provide context-based suggestions for incorrect words.

This work aims to use the existing OCRs for Hindi and provide a solution for multi-font OCR through an error correction approach. Additionally, it aims to provide an OCR solution for Konkani using the existing Hindi OCR and the error correction approach. This chapter discusses in detailed the three proposed and implemented error correction approaches. Here, initial stage error detection is performed using a dictionary.

This chapter is organized as follows: Section 5.2 provides evaluation metrics used for the proposed three error correction algorithms. Section 5.3 details the performance of existing OCR's for different fonts. The three proposed error correction approaches; Section 5.4 gives details of proposed Weighted Syllable-based Spell Correction (WSSC). Section 5.5 discusses the proposed approach of OCR error correction using MLM-BERT. Section 5.6 explores the utilization of Hindi OCR on Konkani documents and discusses the proposed MB-NSC hybrid OCR error correction approach using MLM-BERT and Norvig Spell Corrector.

5.2 Evaluation Metrics

The ISRI analytic tool was used to calculate the word accuracy during the evaluation process (Rice and Nartker, 2022). The following evaluation metrics are used:

- i **Word Accuracy (WA)**: Word accuracy is the percentage of the words recognized correctly from the total words in the ground truth dataset.
- ii **Word Error Rate (WER)**: WER is the percentage of words incorrectly recognized from the total words in the ground truth dataset. It is also equal to 100 minus word accuracy.
- iii **Character Accuracy (CA)**: Character accuracy is the percentage of the character recognized correctly from the total words in the ground truth dataset.
- iv **Character Error Rate (CER)**: CER is the percentage of characters incorrectly recognized from the total characters in the ground truth dataset. It is also equal to 100 minus character accuracy.

5.3 Performance of OCRs for printed text in different font

OCR	Dataset		
	<i>BM_Aparajita</i>	<i>BM_Nakula</i>	Hindi dataset
Tesseract	65.95%	87.76%	89.29%
Indsenz	53.23%	87.00%	87.84%
e-Aksharayan	13.04%	63.79%	78.37%

Table 5.1: OCRs Word Accuracy for *Aparajita* and *Nakula* fonts.

It was essential to study the performance of existing OCRs on different fonts. The dataset *BM_Aparajita* and *BM_Nakula* were used to study the performance of existing OCR (Indsenz, Tesseract and e-Aksharayan) for same text in different font styles. The accuracy for these OCRs is calculated using the ISRI analytic tools for OCR evaluation (Rice and Nartker, 2022). With this tool's help, OCR's word accuracy is calculated as shown in Table 5.1. It was observed that Tesseract OCR accuracy for *BM_Aparajita* dataset is lower by 21.61% compared to *BM_Nakula* dataset. The accuracy of Indsenz OCR, for dataset *BM_Aparajita* is lower by 34.53% compared to *BM_Nakula* dataset. The accuracy of e-Aksharayan OCR for *BM_Aparajita* dataset is lower compared to *BM_Nakula* by 50.75%. All OCR shows low accuracy for *BM_Aparajita* dataset compared to *BM_Nakula* dataset. These existing OCRs also were tested with standard Hindi dataset (Mathew et al., 2016). The obtained accuracy for Tesseract OCR with standard Hindi dataset is 89.29%. The Indsenz OCR accuracy for the standard Hindi dataset is 87.84%, and e-Aksharayan shows an accuracy of 78.37%. Tesseract OCR performs better on *BM_Aparajita* dataset compared to Indsenz and e-Aksharayan. It is clear from Table 5.1 that printed text in different fonts affects the accuracy of the OCR. Among these OCR's, the open-source Tesseract Hindi OCR showed acceptable word accuracy compared to other available OCRs nearly close to Indic OCR (Mathew et al., 2016). On error analysis, three types of errors were observed insertion, substitution, and deletion. Examples of the same is shown in Table 5.2. It has been observed that the misclassification of words varies across different OCR outputs. Also, it was observed that recognized text differs for different fonts. Thus, to handle substitution errors, the Error Correction with Weighted Syllable-based Spell Correction (WSSC) approach was proposed and implemented as discussed in Section 5.4.

Type of error	Correct Word	OCRed Word
Substitution	लेखक	लैखक
Insertion	गोंयचो	गोंयचों
Deletion	मळार	मार

Table 5.2: Example of Substitution, Insertion and Deletion errors.

5.4 Error Correction with Weighted Syllable based Spell Correction (WSSC)

5.4.1 Structure of proposed solution

In this method, a syllable (akshara) confusion matrix was generated, taking into account the frequency of each syllable substituted, creating a frequency-based error mapping file (refer Section 5.4.2). Word was looked up in the dictionary to check whether it is correct or incorrect (refer Section 5.4.3.1). If the word is incorrect, then based on the confusion matrix, appropriate syllables are substituted for all syllables in the words. It forms many words and assigns the weight to words based on the frequency of syllables. Again, these words were looked up in the dictionary for their correct combination, suggesting the best word based on the weight assigned (refer Section 5.4.3.2). To handle new errors, the frequency-based error mapping tuple needs to be updated, thereby demonstrating its robustness

5.4.2 Creation of Frequency based Error Mapping Tuple (*SY_ERR_MAP*)

Analysis of the performance of OCRs accuracy on printed text in different font style, to find the reason for the low accuracy of all the OCRs for printed text in *Aparajita* font, ie *BM_Aparajita* compared to the *BM_Nakula* dataset was carried out. Upon analysis, the errors were categorized into three types: substitution, deletion, and insertion errors. The substitution error details the incorrectly recognized character by OCR, along with the actual character that should have been recognized by the OCR. These details of substitution errors were utilized, to calculate the frequency of each substitution errors based on the occurrence of the same error. These were

Term (Notation)	Meaning
S	Set of Syllables
W	Set of Words or Combination of Syllables
W_V	$W_V \in W$ is a set of valid words which are present Word frequency list (cfilt.iitb, 2023).
W_R	$W_R \in W$, $W_R \in \mathbb{R}$, is a set of OCR recognised words.
W_S	$W_S \in W$, is a most appropriate suggestion for W_R word.
I_i	$I_i \in S$ this is a incorrect recognised syllable present in word.
C_i	$C_i \in S$ this is a correct substitution for I_i recognised syllable present in word.
F_{I_i, C_i}	F_{I_i, C_i} is the frequency for C_i a correct substitution for I_i recognised syllable present in word.
$E = \langle I_i, C_i, F_{I_i, C_i} \rangle$	This denotes the error analysis present in tuple SY_ERR_MAP .
Input_File	It is input file which has recognised output for a given text in image.
Output_File	Output file in which output of WSSC will be stored.

Table 5.3: Terminology and Notation used

stored in Frequency-based Error Mapping Tuple named $SY_ERR_MAP = (I_i, C_i, F_{I_i, C_i})$. These terminology are defined in Table 5.3.

Sr. No.	Recognised Syllable (I_i)	Correct Syllable (C_i)	Frequency (I_i, C_i) (F_{I_i, C_i})
1	लै	ले	50
2	लै	लें	30
3	ख	ख	31
4	ख	स्व	27

Table 5.4: Creation of Frequency-based Error Mapping Tuple (SY_ERR_MAP)

Sr. No.	Correct Word	OCRed Word
1	लेखक	लैखक
2	पड़ने	पडने
3	रोज़	रौज़
4	हौकरी	हाँकरी
5	रोटी	यैटी

Table 5.5: An example of correct word and OCRed word (output of Tesseract OCR).

This tuple $SY_ERR_MAP = (I_i, C_i, F_{I_i, C_i})$ consist of three columns. The first column has an incorrect syllable recognized by OCR represented as I_i . The second column has correct syllable named as C_i which was supposed to be recognized by OCR. The Third column has the frequency of both I_i and C_i occurring together named as F_{I_i, C_i} (see Table 5.4). This error mapping tuple represents errors occurred due to the substitution of similar-looking characters. This is seen in the example as shown in Table 5.5 where two syllables are recognized as similar-looking characters. Example the word रोटी [roti] was recognized as रैटी [jaiti].

5.4.3 Weighted Syllable based Spell Correction Approach

The proposed method, Weighted Syllable based Spell Correction Approach (WSSC), detects the erroneous word from the recognized word W_R and suggests the correct word W_S for the same. The following are major steps in the WSSC approach, whose pseudo-code is mentioned in the algorithm 2.

1. The first step in WSSC approach is to detect errors, as explained in Section 5.4.3.1
2. Secondly, it corrects the detected erroneous words from the recognized output. The following steps it follows, which are explained in detail in the section 5.4.3.2
 - (a) Words Suggestion using error mapping tuple SY_ERR_MAP as mention in Section 5.4.3.3
 - (b) Best Word is suggested using algorithm 3 "Best Suggestion" as discussed in Section 5.4.3.4

5.4.3.1 Error Detection

Error detection is performed by checking if the recognized word W_R belongs to the special character or numeric set. Further, it checks if the recognized word W_R belongs to a valid set of words W_V . In this case, the word W_R is considered correct. On satisfying the condition W_R is written to the *Output_File*. In case these conditions are not satisfied, it suggests the word W_S for recognized word W_R as mentioned in the error correction method in Section 5.4.3.2

5.4.3.2 Error Correction

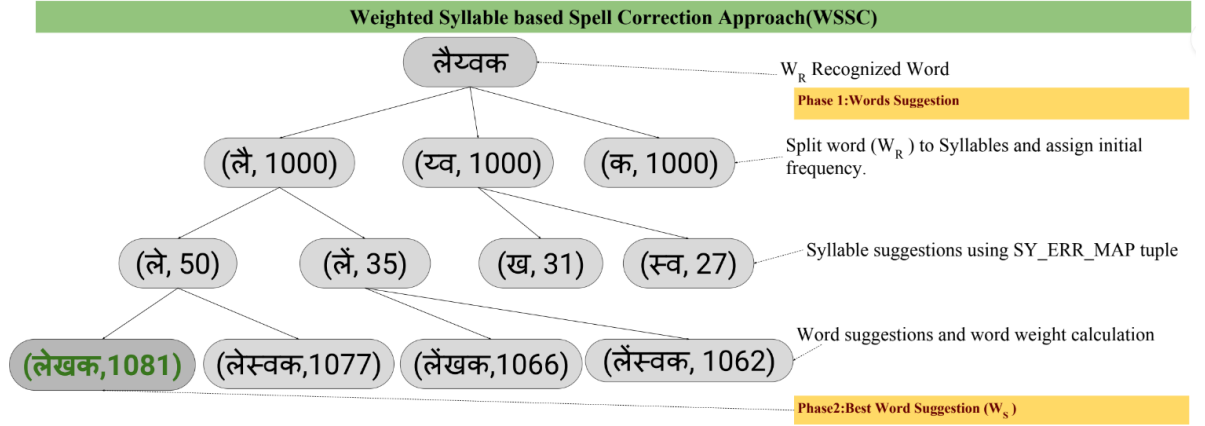


Figure 5.1: An example for error correction steps in WSSC.

The error correction for an incorrectly recognized word W_R , is performed by suggesting all possible words for a recognized word as explained in Section 5.4.3.3. Finally, it suggests the best word W_S from the suggested words for a recognized word W_R , using the Algorithm 3, discussed in Section 5.4.3.4. This procedure for error correction is explained with the help of a diagram as shown in Figure 5.1.

5.4.3.3 Words Suggestion using error mapping tuple SY_ERR_MAP

The words Suggestions algorithm is divided into two parts, the first part is *Syllable Suggestion*, which uses the error mapping tuple *SY_ERR_MAP* to suggested syllable list. The second part is *Words Suggestion*, which uses the suggested syllables list for recommending a likely list of words.

5.4.3.3.1 Syllable Suggestion The following are the steps for suggesting syllable and its frequency using information from error mapping tuple *SY_ERR_MAP*.

Step 1: Split the recognized word W_R into syllables.

Step 2: Initialize the frequency of each syllable with the highest frequency occurring in the tuple *SY_ERR_MAP*.

Step 3: For each of this syllable I_i , appropriate syllable C_i based on the error mapping tuple SY_ERR_MAP was suggested. Further, assigning the corresponding frequency F_i to the syllable C_i , that has been replaced with syllable I_i .

Repeat step 3 for all the syllables in the recognized word W_R . Finally, this procedure returns the list of syllables C_i along with its frequency F_i . This combination of C_i , F_i is the possible mapping of I_i as shown in Figure 5.1.

Sug ← **Sug_Words**(**WordAsSyl**[], **SSL**, **Index**, **Freq**[])

/ WordAsSyl: Word stored as syllable array.*

SSL: Suggested Syllables C_i and Frequency F_i

Index: C_i is substituted at this position

Freq: Frequency of syllable for WordAsSyl.

*Sug: Returns a Class name as Suggests with data member word and freq. word contain the suggested word and freq which stores frequency of syllables for the suggested word. */*

Input: WordAsSyl[], SSL, Index, Freq[]

Output: Sug(words, Freq)

```

1: Sug(words, Freq) = empty
2: for  $C_i, F_i$  in SSL do
3:   temp_words[] ← WordAsSyl[] /*copy WordAsSyl to temp
4:   temp_words[Index] =  $C_i$ 
5:   temp_Freq = Freq
6:   temp_Freq[Index] =  $F_i$ 
7:   Sug.append(temp_words, temp_Freq)
8: end for
9: return Sug

```

Algorithm 1: Words Suggestion

5.4.3.3.2 Words Suggestion On obtaining the correct syllable C_i and its corresponding frequency F_i for syllable I_i , for each of the syllables in erroneous word W_R . Thus, using the syllable C_i and its frequency F_i for suggesting words as shown in Figure 5.1 and mentioned in the Algorithm 1. In this algorithm, it replaces each of the suggested syllable C_i in the position of the

actual syllable I_i in the word W_R . On substituting the syllable at a particular position, it modifies the frequency of that syllable with the frequency F_i of the replacing syllable C_i , thus forming a new word, which was stored in the suggestion list as shown in Figure 5.1. The same was repeated for each of the syllable in syllable suggested list. The word suggestion list also consists of an array of frequency for each of the syllables, in the word.

5.4.3.4 Best Word Suggestion

After obtaining all the suggested words, which is the output of word the suggestion procedure, to choose the best word from the suggested word, it is checked if any one of the suggested words is present in word list W_V and then it writes that particular word to *Output_file*. If the word does not appear in the word valid list W_V then the algorithm suggest words W_S that has the highest weight. The pseudo-code is mention in the algorithm 3. The weight of the word is the sum of the frequency of syllables in the recognized W_R . This is written to the output file *Output_File*. The Same WSSC approach is repeated for all the word W_R in the input file *Input_File*.

5.4.4 Results and Analysis

For this experiment, the test dataset used are *BM_Aparajita* and *BM_Nakula* as created in Section 4.2.1 (refer Chapter 4). The WSSC approach detects errors purely based on the word present in the W_V word list. Further, if the word is not listed in W_V , then it suggests words based on WSSC approach. WSSC approach was tested on the Tesseract, Indsenz and e-Aksharayan OCR output. This approach has increased the OCR accuracy nearly by 2.3% as shown in Table 5.6. And on testing the proposed WSSC approach with *BM_Nakula* dataset and it is been observed that it has improved accuracy of Tesseract OCR by small margin 0.50% and the accuracy for Indsenz OCR is increased by 0.23% as shown in Table 5.6.

Table 5.7 shows the second column contains the correct word, the third column has the OCR'd word, and the fourth column has the words suggested by the WSSC approach. However, this approach does not handle insertion and deletion errors.

Handling insertion and deletion errors through post-processing approaches is challenging, but these errors can be effectively addressed with the help of contextual information. Some of the cases that can be handled using context information are indicated in Table 5.8. The red color

Input: Input_File (text file ie output of OCR)

Output: Output_File (text file)

```

1: for  $W_R$  in Input_File do
2:   if Is_Spl_Char( $W_R$ ) or Is_Num( $W_R$ ) or  $W_R \in W_V$  then
3:     Write  $W_R$  to Output_File.
4:   else
5:     Suggested_words[: List<String[]>; Modified_freq[: List<String[]>; Max=0: Int
6:     Syl_List[: String[] /*WordToSyllable generate grapheme clusters for string  $W_R$ */
7:     Syl_List[]  $\leftarrow$  WordToSyllable( $W_R$ )
8:     freq[]  $\leftarrow$  len(Syl_List): init to 1000
9:     for  $I_k$  in Syl_List do
10:      Sug_Syl SSL : Sug_Syl (C[], F[])
11:      /*Suggest_Syl( $I_k$ ) returns ( $C_i, F_i$ )  $\forall I_k \in I_i$  from SY_ERR_MAP */
12:      SSL( $C_i, F_i$ )  $\leftarrow$  Suggest_Syl( $I_k$ )
13:      prev_sug_word = Suggested_word
14:      prev_freq  $\leftarrow$  Modified_freq
15:      sug  $\leftarrow$  Sug_Words(Syl_List, SSL, k, freq)
16:      for  $s_l$  in sug do
17:        Suggested_words.append( $s_l$ .word); Modified_freq.append( $s_l$ .freq)
18:      end for
19:      for  $sw_m$  in prev_sug_word do
20:        sug  $\leftarrow$  Sug_Words( $sw_m$ , SSL, m, prev_freq[m])
21:        for  $s_l$  in sug do
22:          Suggested_words.append( $s_l$ .word); Modified_freq.append( $s_l$ .freq)
23:        end for
24:      end for
25:    end for
26:     $W_S$  = SuggestBestWord(Suggested_words, Modified_freq)
27:    Write  $W_S$  to Output_File.
28:  end if
29: end for

```

Algorithm 2: Weighted Syllable based Spell Correction Approach (WSSC)

word=SuggestBestWord(Suggested_words,Modified_freq)

Input: Suggested_words, Modified_freq

Output: Word

```

1: BestIndex=0: Int
2: Flag=False
3: for  $w_l$  in Suggested_words do
4:   /* Calculates the sum of frequency */
5:   Sum  $\leftarrow$  Cal_Sum(Modified_freq[l])
6:   if Sum > Max then
7:     BestIndex  $\leftarrow$  l
8:     Max  $\leftarrow$  Sum
9:   end if
10:  if  $w_l \in W_V$  then
11:    Word  $\leftarrow$   $w_l$ 
12:    Flag  $\leftarrow$  True
13:    break
14:  end if
15: end for
16: if Flag = True then
17:   Word  $\leftarrow$  Suggested_words[BestIndex]
18: end if
19: return Word

```

Algorithm 3: Best Word Suggestion

Dataset	Output	Word accuracy		
		Tesseract	Indsenz	e-Aksharayan
<i>BM_Aparajita</i>	OCR	65.95%	53.23%	13.04%
	WSSC	67.19%	88.00%	13.39%
<i>BM_Nakula</i>	OCR	87.76%	87.00%	63.79%
	WSSC	88.22%	87.23%	64.70%

Table 5.6: Word accuracy obtained using WSSC

Sr. No.	Correct Word	OCRed Word	WSSC Output
1	लेखक	लैखक	लेखक
2	पढ़ने	पडने	पढ़ने
3	रोज़	रौज़	रोज़
4	हौकरी	हाँकरी	हौकरी
5	रोटी	यैटी	रोटी
6	वगैरा	वगैरग	वगैरग

Table 5.7: Output of WSSC

e.g. 1	Actual text (Hindi)	उत्पादों को पेश कर रहे हैं।
	Transliteration (English)	<i>UTPADOM KO PESa CaRa HAI.</i>
	Translation (English)	Presenting products.
	OCR recognized text	उत्पादों का पेश कर रहे हैं
e.g. 2	Actual text (Hindi)	वायसराय के अंगरक्षक के रूप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।
	Transliteration (English)	<i>VAYaSaRAYa KE aMGaRaKSaKa KE RuPa mEM GaThita YaHa BhARatIYa SENA KA SaBaSE PuRANA REJImEMTa HAI.</i>
	Translation (English)	Formed as the bodyguard of the Viceroy, it is the oldest regiment of the Indian Army.
	OCR Recognized text	वायसराय के अंगरक्षक के रूप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।
e.g. 3	Actual text (Hindi)	6 हजार नौकरियों का सृजन होगा।
	Transliteration (English)	<i>6 HaJARa NAuCaRIYOM CA SRUJaNa HOGA.</i>
	Translation (English)	6 thousand jobs will be created.
	OCR recognized text	ह हजार नौकरियों का सृजन होगा।

Table 5.8: Output of *Hindi_Error_Correction_Model* indicating suitable output as per context.

indicates an incorrect word and green indicates correct word. Example 1 where the OCR'd word का [kã] is incorrect. In this case, the existing bigram model can show suggestions as का [ka], के [ke], की [ki], etc. Likewise, in example 2, the OCR'd word रनप [rənəp] is an incorrect word. Existing models can suggest a word like बाद [baɖə], साथ [saṭṭhə], लिए [lie] etc. Similarly, in example 3 the incorrect word is ह, which is incorrect word. For such cases, an approach is needed that can suggest a word based on context. However, the existing approaches do not provide suggestions based on context (refer Chapter 2; Section 2.3.2). From this analysis, we proposed and implemented an approach that can mask erroneous words and provide suggestions based on contextual information (as presented in Section 5.5).

5.5 Error Correction using Masked-Language Modeling With Bidirectional Encoder Representations from Transformers (MLM-BERT)

This section is organized as follows; Section 5.5.1 provides the structure of proposed solution. Section 5.5.2 provides details of dataset used for the experiment. Section 5.5.3 provides the details of proposed error correction using MLM-BERT. Section 5.5.4 discusses the results.

5.5.1 Structure of proposed solution

This approach mainly focuses on error correction. In the initial phase, errors are detected using a dictionary. A BERT tokenizer and model built for Hindi are then used to correct errors and provide contextual suggestions. The erroneous word is masked with the token [MASK], allowing the model to provide suggestions for the masked word based on the available context. This is tested using Hindi data.

5.5.2 Datasets

The training data used for this experiment is shown in Table 5.9. The test dataset is *Hindi_test_dataset* and *Hindi-IITH-dataset* created in Section 4.3.3 (refer Chapter 4). The characteristics of test

dataset is shown in Table 4.2 (refer Chapter 4; Section 4.3.3).

Language	Total Words	Source of data
Hindi	8215774	Hindi dataset (Shivam, 2022)

Table 5.9: Training Data used to create *Dev-Hindi* tokenizer and *Hindi_Error_Correction_Model*

5.5.3 Experimental setup for Error correction using MLM-BERT

5.5.3.0.1 Proposed procedure *error-correction-mlm-bert* The proposed system consisting of three phases: I. Training BERT tokenizer, II. Training MLM BERT model and III. Error Correction. For convenience it is named as *error-correction-mlm-bert* procedure. The below is the detailed explanation of each phase involved in the *error-correction-mlm-bert* procedure:

I. **Training BERT tokenizer:** The BERT model requires the input data in a suitable format to obtain the corresponding embedding of the sentence. The input to the BERT model is a sentence of a fixed length. Each token was given a distinct ID when the BERT model was trained. So, in order to employ a pre-trained BERT model, each token in the input sentence must first be converted into its corresponding unique ID. Sentences are padded to achieve a fixed length for input. And to obtain a unique ID for a token a BERT tokenizer is needed. The BERT tokenizer will split the word into two subwords (token). The first token is frequently seen word (prefix) in a corpus. Two hashes are prefixed to the second token to indicate that it is a suffix following some other subwords. Further, it converts subwords to unique IDs using the BERT tokenizer.

The BERT tokenizer for any language *<language>* can be created using BertWordPieceTokenizer¹ and the respective data *<training-data>*, which is named as *<language>* tokenizer.

II. **Training MLM BERT model:** Here the BertForMaskedLM architecture is employed specifically utilizing the BERT Model known as "*bert-base-uncased*". This particular

¹https://github.com/huggingface/tokenizers/blob/main/bindings/python/examples/train_bert_wordpiece.py

model consists of 12 layers, 768 hidden units, 12 attention heads, and a total of 110 million parameters. It was trained on lowercase English text, which works well for English but not for Indian languages. To support Indian language text the BERT *<language>* tokenizer as created in phase I, is used and a corpus for additional training of the pre-trained BERT model "bert-base-uncased" is also utilized, which is the same as the one used during phase I for constructing the tokenizer. The training parameters should be configured i.e. hidden activation function: *<hidden-activation-function>*, the maximum position embeddings *<max-pos-embedding>*, the attention dropout rate *<dropout-rate>*, the number of epochs *<number-epochs>*, the instantaneous batch size per device *<batch-size>*, the gradient accumulation steps were set to *<gradient-steps>*. On training it results in creating model referred as *<language>* model.

III. Error Correction: Initially the lookup dictionary is utilized to identify incorrect words. And the word correction was achieved by utilizing the *<language>* model and its corresponding tokenizer i.e. *<language>*. First, the input OCR text is tokenized using the *<language>* tokenizer, resulting in token IDs. Subsequently, the erroneous word is masked using the special token [MASK], and the *<language>* model generates a set of predictions for the word in the form of token IDs and their associated probabilities. These token IDs are then converted back to words using the *<language>* tokenizer. BERT generates context-sensitive embeddings by looking at both the left and right context of each word in a sentence. Employing the Levenshtein distance for the top 6 candidate words, the similarity between the original token and the predicted token was assessed by selecting those having a similarity score of more than 0.8. Ultimately, the word with the highest probability among the suggested options is selected by this approach as most suitable word.

The procedure *error-correction-mlm-bert* (refer Section 5.5.3.0.1) is used to create the BERT tokenizer and BERT model for Hindi. Figure 5.2 shows the block diagram of proposed system of error correction using MLM-BERT. Firstly it creates the BERT tokenizer for Hindi using *error-correction-mlm-bert* Phase I and the training data (refer Table 5.9) This tokenizer is named as *Dev-Hindi* tokenizer Secondly it creates the BERT model using *error-correction-mlm-bert* Phase II, the *Dev-Hindi* tokenizer and training data (refer Table 5.9). The the training parameters were configured to create BERT model are as follows: Gaussian Error Linear Unit (GELU) as

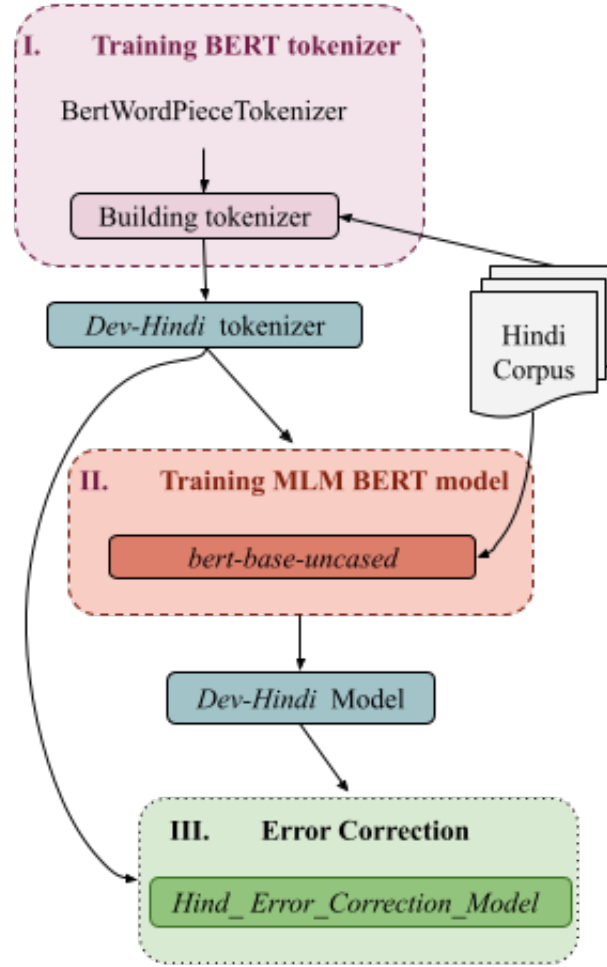


Figure 5.2: Proposed system of Error correction using MLM-BERT

hidden activation function, the maximum position embeddings were limited to 512, the attention dropout rate was set to 0.1, the number of epochs was set to 3, the instantaneous batch size per device was 8, the gradient accumulation steps were set to 1, the number of sentences was 539,134 and the total number of optimization steps was 50,500. The GELU was chosen as an activation function as it is a high-performing neural network activation function (Hendrycks and Gimpel, 2016). The choice of dropout rate and epoch configuration was influenced by a study demonstrating good performance results in the context of BERT (Sun et al., 2019). After training the training loss was 1.54. It creates the model for Hindi named as *Dev-Hindi* model.

Thirdly, *error-correction-mlm-bert* Phase III. is used for suggesting word for every incorrect word using the *Dev-Hindi* tokenizer and *Dev-Hindi* model. This model is named as *Hindi_Error_Correction_Model*. Here intermediate stage suggests top 6 candidate words along with probability. Finally the word with the highest probability among the suggested options is output of *error-correction-mlm-bert*.

Phase III. as most suitable word. An example is shown in Table 5.10. The word selected by system can be either correct, or incorrect. In some cases, the system may not provide any suggestions, and in those instances, the word remains unchanged. Example for each of these cases are discussed in the section 5.5.4. The correct words are indicated in green, the incorrect words are underlined and marked in red and the words for which no suggestions were provided are indicated in blue.

Actual word	OCRed text as input	suggestion by Model*	Probability	Model* output
नई दिल्ली	<u>राई</u> दिल्ली	नई	0.06	नई दिल्ली
		आज	0.04	
		आईआईटी	0.02	
		नयी	0.014	
		राजधानी	0.012	
		पुरानी	0.008	
*Hindi_Error_Correction_Model				

Table 5.10: Example of *Hindi_Error_Correction_Model* output Hindi language.

5.5.4 Results and Analysis

The proposed system has been evaluated with *Hin_test_dataset* and *Hindi-IIITH-dataset*, whose characteristics are mentioned in Table 4.3 (refer Chapter 4; Section 4.3.3) using the ISRI analytic tool for OCR evaluation (Rice and Nartker, 2022). Word accuracy and Word Error Rate (WER) are used as evaluation metrics. It was observed that Tesseract Hindi OCR word accuracy on *Hin_test_dataset* and *Hindi-IIITH-dataset* were 87.23% and 89.29%, respectively. And the word accuracy achieved by *Hindi_Error_detection_Model* on the dataset *Hin_test_dataset* and *Hindi-IIITH-dataset* were 89.65% and 95.18% respectively, as shown in Table 5.11. Therefore, it can be observed that there is an improvement in word accuracy on both testing dataset *Hin_test_dataset* and *Hindi-IIITH-dataset*. The difference in word accuracy between the Tesseract OCR results

obtained from the *Hin_test_dataset* and the *Hindi-IIITH-dataset* has been noticed. It is observed the increase in word accuracy from 91.60% to 95.18%, having an improvement of 3.58% over the baseline Hindi OCR word accuracy (Mathew et al., 2016).

Dataset	Word accuracy	
	Tesseract OCR	<i>Hindi_Error_Correction_Model</i>
<i>Hin_test_dataset</i>	87.23%	89.65%
<i>Hindi-IIITH-dataset</i>	89.29%	95.18%

Table 5.11: Word accuracy of Tesseract OCR and *Hindi_Error_Correction_Model*

Table 5.12 shows the category-wise WER for the output of the Tesseract OCR and *Hindi_Error_Correction_Model*. It is observed that *Hindi-IIITH-dataset* had more errors in words without any modifiers. This is because of the existence of punctuation mark in the *Hindi-IIITH-dataset* which is constantly misclassified by Tesseract OCR output. The proposed error correction approach corrects it, indicating proper punctuation marks for all the instances of the sentence. Thus, the *Hindi_Error_Correction_Model* corrects most of the words that have no modifiers. Further, it is observed that there is a decrease in error for words with *Ekar*, *Ookar*, *Conjunct consonants*, and *words with multiple modifiers*. Words with *anusvara* error rate have remained the same. However it is important to note that these error words were masked and the *Hindi_Error_Correction_Model*'s correction is based more on contextual information.

The results of this research work indicate that the proposed model delivers accurate recommendations, as demonstrated by the data presented in Table 5.13. For e.g the actual word को [ko] is recognised as को [kã] via tesseract OCR systems whereas the proposed model corrects it to को [ko]. Let us consider another example to illustrate the effectiveness of the proposed model: The actual word बधाई [bəd̪h̪aɪ] is recognized by OCR as वाई [vai:] while the correction provided by the suggested model is बधाई. Additionally, the proposed model exhibits contextual-based corrections, as evidenced by the information provided in Table 5.14 and Table 5.15. For example the actual text 6 हजार [ts̪h̪əhə həd̪zar] was incorrectly recognised as ६ हजार [hə həd̪zar] and the proposed model suggests 10 हजार [d̪əs həd̪zar]. However the suggested word is incorrect, interesting is that it suggested number and not any other word close to ६ [hə]. The Model also provides incorrect suggestions including cases where incorrect suggestions are given based on the context, as illustrated in Table 5.16. However the suggested word is very similar to the correct word

Word category	Dataset			
	<i>Hin_test_dataset</i>		<i>Hindi-IIITH-dataset</i>	
	Tesseract	Model*	Tesseract	Model*
	Hindi OCR WER %	WER %	Hindi OCR WER %	WER %
<i>Without modifiers</i>	1.38	0.45	3.5	0.67
<i>Matras</i>	2.11	1.35	2.14	1.02
<i>Eekar</i>	0.96	0.82	0.92	0.75
<i>Ookars</i>	0.13	0.12	0.14	0.12
<i>Conjunct consonant</i>	0.99	0.87	0.98	0.5
<i>Anusvara</i>	0.68	0.68	0.65	0.65
<i>Candra</i>	0.05	0.05	0.03	0.03
<i>Words with multiple modifiers</i>	6.47	6.01	2.35	1.08
Total WER	12.77	10.35	10.71	4.82
<i>*Hindi_Error_Correction_Model</i>				

Table 5.12: Word error rate(WER) of *Hindi_Error_Correction_Model* testing on *Hin_test_dataset* and *Hindi-IIITH-dataset* for each Category of words

Actual word	OCRed output	Model* output	Actual word	OCRed output	Model*
को	<u>कां</u>	को	बधाई	<u>वाई</u>	बधाई
और	<u>ओंर</u>	और	इंजीनियरिंग	<u>इंजीनियरिंग</u>	इंजीनियरिंग
इन	<u>हन</u>	इन	दौरा	<u>दोस</u>	दौरा
कहा	<u>केहा</u>	कहा	प्रतिनिधित्व	<u>प्रतिनिधित्व</u>	प्रतिनिधित्व
इजरायल	<u>इजरायल</u>	इजरायल	साथसाथ	<u>साथस्नाथ</u>	साथसाथ
सीमाओं	<u>सीमाओं</u>	सीमाओं	इस	<u>हस</u>	इस
इंजीनियरिंग	<u>इंजीनियरिंग</u>	इंजीनियरिंग	पर	<u>प्रर</u>	पर
<i>*Hindi_Error_Correction_Model</i>					

Table 5.13: An example of correct suggestion provided by *Hindi_Error_Correction_Model*

Actual text (Hindi)	उत्पादों को पेश कर रहे हैं।
Transliteration (English)	<i>UTPADOM KO PESa CaRa HAI.</i>
Translation (English)	Presenting products.
OCRed output	उत्पादों का पेश कर रहे हैं
<i>Hindi_Error_Correction_Model</i> Output	उत्पादों को पेश कर रहे हैं
Actual text (Hindi)	वायसराय के अंगरक्षक के रूप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।
Transliteration (English)	<i>VAYaSaRAYa KE aMGaRaKSaKa KE RuPa mEM GaThita YaHa BhARatIYa SENA KA SaBaSE PuRANA REJImEMTa HAI.</i>
Translation (English)	Formed as the bodyguard of the Viceroy, it is the oldest regiment of the Indian Army.
OCRed output	वायसराय के अंगरक्षक के रूप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।
<i>Hindi_Error_Correction_Model</i> Output	वायसराय के अंगरक्षक के रूप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।
Actual text (Hindi)	6 हजार नौकरियों का सृजन होगा।
Transliteration (English)	<i>6 HaJARa NAuCaRIYOM CA SRUJaNa HOGA.</i>
Translation (English)	6 thousand jobs will be created.
OCRed output	ह हजार नौकरियों का सृजन होगा।
<i>Hindi_Error_Correction_Model</i> Output	10 हजार नौकरियों का सृजन होगा।

Table 5.14: Output of *Hindi_Error_Correction_Model* indicating suitable output as per context.

or contextually appropriate. Here is an example of the observation, i.e the actual word विचारों [vɪʃarõ] and OCR recognised word as विचारो [vɪʃaro] and *Hindi_Error_Correction_Model* suggestion is विचारो [vɪʃaro] which is a quite similar suggestion to actual word. Another example for similar suggestion is with actual word उद्घाटन [ud̪gʰaʈən], OCR-recognized word उद्घाटन [ud̪gʰaʈən] and proposed model suggestion is उदघाटन [ud̪əgʰaʈən]. Furthermore, an instance that demonstrates contextual-based recommendations is when real words is इसे ([ise],

OCR recognised word is हरने [hərəne] and model suggested the word as महामहिम [məhaməhim], where the system suggested word was suitable for the sentence as shown in Table 5.17 and cases where suggestions were not offered for words are depicted in Table 5.18 and Table 5.19. For e.g having actual word हिस्सा [hissa] and OCR recognised word द्विरुसा [d̥vīrusa], the proposed model did not provide any suggestion for word द्विरुसा [d̥vīrusa], result in retaining same word. However, when the majority of neighbouring words were incorrect, the model fails to provide appropriate suggestions, as shown in Table 5.20.

Actual text (Hindi)	पोत परिवहन मंत्रालय
Transliteration (English)	POTa PaRIVaHaNa maMTRALaYa.
Translation (English)	Ministry of Shipping
OCRed output	गोल परिवहन मंत्रालय
Hindi_Error_Correction_Model Output	पोत परिवहन मंत्रालय
Actual text (Hindi)	एफटीआईआई की वर्तमान छात्रा
Transliteration (English)	EFaTIAI KI VaRTaMANa ChATRA
Translation (English)	Current student of FTII
OCRed output	एफटीआईआई की वर्तमान छात्रा
Hindi_Error_Correction_Model Output	एफटीआईआई की वर्तमान छात्रा

Table 5.15: Output of *Hindi_Error_Correction_Model* suggest correct suggestions for Noun.

5.6 Error correction with the hybrid approach: MLM-BERT and Norvig Spell Corrector (MB-NSC)

This section is organized as follows: Section 5.6.1 discusses the motivation for this work. Section 5.6.2 provides the structure of the proposed solution. Section 5.6.3 provides details of the experiment setup to study the performance of Tesseract Hindi OCR on the Konkani dataset. Section 5.6.4 provides the experiment setup for the proposed error correction MB-NSC approach. Section 5.6.5 discusses the performance of Tesseract Hindi OCR on Konkani documents. Section 5.6.6 discusses the result of MB-NSC approach on Konkani data.

Actual word	OCRed output	Model* output	Actual word	OCRed output	Model* output
महत्व	<u>महत्त्व</u>	<u>महत्त्व</u>	उद्धाटन	<u>उद्धाटन</u>	<u>उदघाटन</u>
उप	<u>उप्र</u>	<u>भारतीय</u>	विचारों	<u>विचारो</u>	<u>विचारो</u>
व्यवहार	<u>हयवहार</u>	<u>हयवहार</u>	आयोजन	<u>प्रायोजना</u>	<u>परियोजना</u>
गांवों	<u>मांओं</u>	<u>भारत</u>	संयुक्त	<u>सयुक्त</u>	<u>सयुक्त</u>
केंद्रीय	<u>केद्रीय</u>	<u>केद्रीय</u>	दिवस	<u>दिवा</u>	<u>दिवा</u>
दिल्ली	<u>दिल्लो</u>	<u>दिल्लो</u>	जीवंत	<u>जीवत</u>	<u>जीवत</u>
*Hindi_Error_Correction_Model					

Table 5.16: Output of *Hindi_Error_Correction_Model* having quite similar suggestion to actual word or appropriate based on context or word.

Actual text (Hindi)	27 जनवरी 1950 को इसे राष्ट्रपति का अंगरक्षक नाम दिया गया।
Transliteration (English)	27 JaNaVaRI 1950 KO ISE RAsRaPaTI KA aMGaRaKsaKa NAm DIYA GaYA.
Translation (English)	On 27 January 1950, it was named the President's Bodyguard.
OCRed output	<u>17</u> जनवरी 1950 को <u>हरने</u> राष्ट्रपति का अंगरक्षक नाम दिया गया।
<i>Hindi_Error_Correction_Model</i> Output	<u>17</u> जनवरी 1950 को <u>महामहिम</u> राष्ट्रपति का अंगरक्षक नाम दिया गया।

Table 5.17: Output of *Hindi_Error_Correction_Model* Although incorrect suggestion, but valid based on context.

5.6.1 Motivation

The error correction approach using MLM-BERT provided suggested words that were appropriate based on context. There were incorrect words suggested by the model but contextually appropriate. Nevertheless, the incorrect suggestions often align closely with the actual word or most likely based on context. It was also observed that in some cases, the model does not provide any suggestions. These discrepancies may arise due to incorrect consecutive words or contextual

Actual text (Hindi)	2008 में
Transliteration (English)	2008 mEM
Translation (English)	In 2008
OCR recognized text	<u>1008</u> में
<i>Hindi_Error_Correction_Model</i> Output	1008 में

Table 5.18: Output of *Hindi_Error_Correction_Model* provided no suggestion

Actual word	OCRed output	Model* output	Actual word	OCRed output	Model* output
हिस्सा	<u>द्विरुसा</u>	द्विरुसा	रिजिजू	<u>रिज्जिजू</u>	रिज्जिजू
रस्मी	<u>ररब्बेमी</u>	ररब्बेमी	सेना	<u>गोया</u>	गोया
सुविधा	<u>सुबिधा</u>	सुबिधा	स्वागत	<u>रत्रागत</u>	रत्रागत
चेंगप्पा	<u>चेंगप्या</u>	चेंगप्या	प्रणाली	<u>प्याली</u>	प्याली
रुपये	<u>रुपर्टा</u>	रुपर्टा	धर्म	<u>घर्म</u>	घर्म
* <i>Hindi_Error_Correction_Model</i>					

Table 5.19: More example of Output of *Hindi_Error_Correction_Model* providing No suggestion.

Actual text (Hindi)	गणमान्य व्यक्ति भी इस अवसर पर उपस्थित थे।
Transliteration (English)	GanamANYa VYaKTI BhI ISa aVaSaRa PaRa UPaSThITa ThE.
Translation (English)	Dignitaries were also present on the occasion.
OCRed output	<u>गणमानय</u> व्यक्ति भी <u>हस</u> <u>अबरार</u> पर उपस्थिति थे।
<i>Hindi_Error_Correction_Model</i> Output	गणमानय व्यक्ति भी <u>शरी</u> <u>अबरार</u> पर उपस्थिति थे।

Table 5.20: Output of *Hindi_Error_Correction_Model* with incorrect words.

considerations. However, sometimes, the suggested word length is two times more or less than the actual word. Moreover, this results in more insertion and deletion errors. Further, working

towards the main objective of this research is to provide an OCR solution for Konkani. Tesseract Hindi OCR does give acceptable accuracy. Thus, this work utilizes the Tesseract Hindi OCR for Konkani documents. Moreover, the proposed MB-NSC approach can improve the accuracy of the Konkani OCR'd text.

5.6.2 Structure of proposed solution

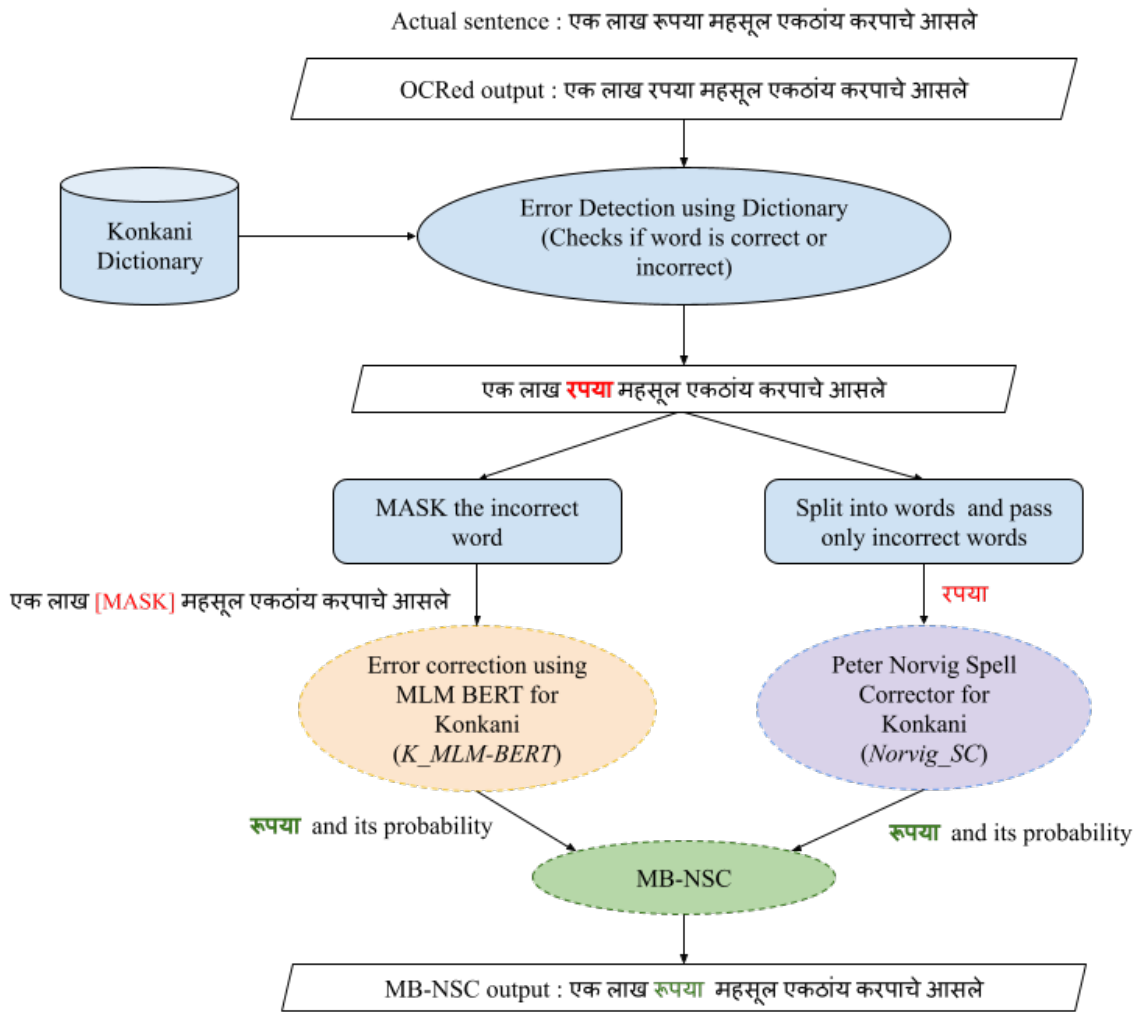


Figure 5.3: Proposed architecture of MB-NSC error correction approach

This section briefly discusses the proposed solution to address the main research objectives, i.e., to provide a solution for Konkani OCR. Thus, exploring the use of the Tesseract Hindi OCR on Konkani documents (refer section 5.6.3). Further, proposed a hybrid error correction approach. The architecture of the hybrid approach is presented in Figure 5.3, and discussed its details in Section 5.6.4 The hybrid error correction approach is designed by combining MLM-BERT

and the Peter Norvig spell corrector (MB-NSC). This will address contextual errors while also identifying the nearest correct word for an incorrect word in the OCR output.

MLM-BERT provide contextual suggestion. The Peter Norvig spell corrector suggests the most likely word for a given input word with the highest probability. Here, the input is the OCR'd word. Combining these two approaches contributes to the novel error correction approach for Konkani OCR'd documents. Given that the proposed approach incorporates contextual suggestions along with identifying the most likely nearest words, it is anticipated that the approach would provide correct word suggestions for sentences having multiple incorrect words.

5.6.3 Experimental setup to study the performance of Tesseract Hindi OCR on *Konkani_OCR_large_test_dataset*

Utilizing a pre-existing Optical Character Recognition (OCR), such as Tesseract Hindi OCR, for a different language like Konkani necessitates comprehensive study. Thus, studying the performance of Tesseract Hindi OCR on a specific dataset, such as the *Konkani_OCR_large_test_dataset* (created in chapter 4; Section 4.3.4.2), involved setting up an experiment to evaluate how accurately existing Tesseract Hindi OCR can recognize text in the Konkani language. The process for setting up the Tesseract OCR involved subsequent steps: Installing the Tesseract OCR tailored for Hindi training data in the Devanagari script and then testing it with the *Konkani_OCR_large_test_dataset*. To evaluate its accuracy in terms of both character and word, the OCR assessment tool ([Rice and Nartker, 2022](#)) was utilized.

5.6.4 Experimental setup for proposed OCR error correction approach: MB-NSC

This section delves into the proposed hybrid automatic OCR error correction using MLM-BERT and Norvig Spell corrector termed 'MB-NSC'. The dataset used for training and testing is detailed in Section 5.6.4.1. This hybrid approach combines two OCR error correction approaches: the first one with MLM BERT for Konkani, discussed in Section 5.6.4.2. The second one employs the Peter Norvig spell corrector for Konkani, detailed provided in Section 5.6.4.3. Section 5.6.4.4 presents the proposed hybrid OCR error correction, i.e., MB-NSC approach. The archi-

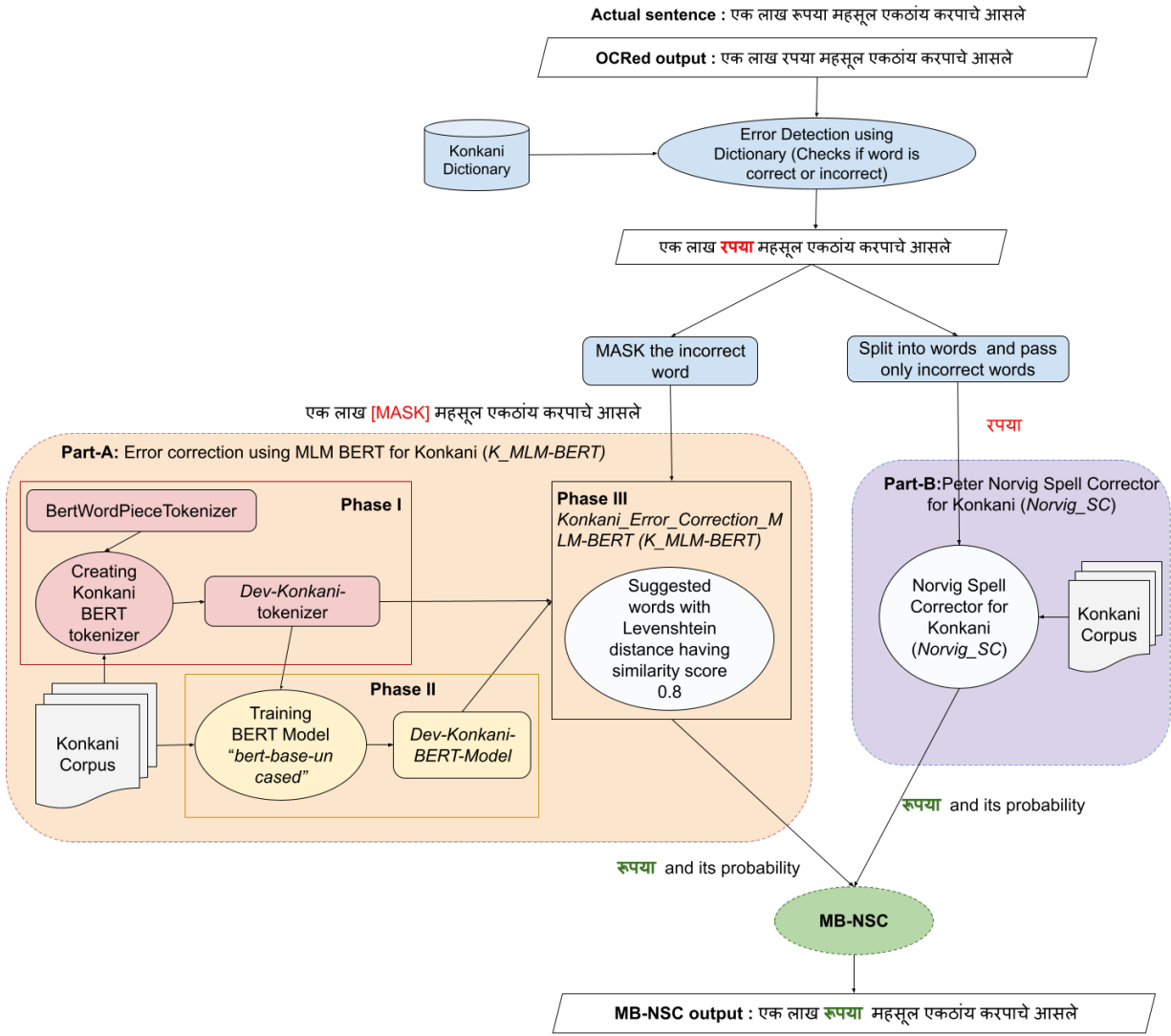


Figure 5.4: Architecture of Proposed MB-NSC error correction approach

ecture of the proposed MB-NSC error correction approach is shown in Figure 5.4. The following subsections illustrate the MB-NSC approach.

5.6.4.1 Datasets

The gold standard Konkani raw text corpus from LDC-IL (Ramamoorthy et al., 2019), whose properties are indicated in sub-section 4.3.4.1, has been used in building and testing our MB-NSC approach. This data has been split into training and test data as shown in Table 5.21. Further test data was used to create these dataset (a) *Konkani_OCR_subset_test_dataset* and (b) *Konkani_EC_test_dataset* as discussed in Chapter 4; Section 4.3.4.3 & section 4.3.4.4.

Language	Words for training	Words for testing	Source of data
Konkani	3591950	27067	Konkani dataset (Ramamoorthy et al., 2019)

Table 5.21: Training and Test data used for Error Correction approach (MB-NSC).

5.6.4.2 Error correction using MLM BERT for Konkani

The employed MLM BERT error correction model has three phases. I) Creating BERT tokenizer for Konkani, II) Creating MLM-BERT model with Konkani corpus, and III) Konkani Error Correction MLM-BERT (*K_MLM-BERT*) as shown in part-A of Figure 5.4. Procedure *error-correction-mlm-bert* (refer Section 5.5.3.0.1) is utilized to create BERT tokenizer and model for Konkani. The details are presented below for each of the phases;

I. Creating BERT tokenizer for Konkani The BERT tokenizer for Konkani was created providing the Konkani training data (see Section 5.21) as input to Procedure *error-correction-mlm-bert*; Phase 1 (refer Section 5.5.3.0.1). This new BERT tokenizer for Konkani was named *Dev-Konkani-tokenizer*.

II. Creating MLM-BERT model with Konkani corpus

Procedure *error-correction-mlm-bert*; Phase II. takes input as *Dev-Konkani-tokenizer* (refer above Phase I.) and training data (see Section 5.6.4.1; Table 5.21) and creates the BERT model for Konkani, to handle Konkani text, named as *Dev-Konkani-BERT-Model*. The following training parameters were set: The maximum position embeddings were calculated using the Gaussian Error Linear Unit (GELU) as the hidden activation function. The maximum number of position embeddings was set to 512, the attention dropout rate was set to 0.1, the total number of epochs was determined as 3, the instantaneous batch size per device was set to 8, the gradient accumulation steps was set to 1, the total number of optimization steps was 40,200. This resulted in the development of *Dev-Konkani-BERT-Model* with training loss 2.235.

III. Konkani Error Correction MLM-BERT (K_MLM-BERT): By utilizing the *Dev-Konkani-BERT-Model* model and its corresponding tokenizer, i.e., *Dev-Konkani-tokenizer*, correct words were suggested with their probability. Employing the Levenshtein distance for the top 6 candidate words, the similarity between the original token and the pre-

dicted token was assessed by selecting those having a similarity score of more than 0.8. Subsequently, the word with the highest probability among the suggested options is selected by the system as the most suitable word. Thus creating Error Correction model *Konkani_Error_Correction_MLM-BERT* (*K_MLM-BERT*) for Konkani. An example is shown in Table 5.22.

Actual word	OCRed word as input	Suggestions by <i>K_MLM-BERT</i>	Probability	Final suggestion by <i>K_MLM-BERT</i>
सरकाराचेर (‘on the government’)	<u>सरकाराचंर</u>	सरकाराचेर (‘on the government’)	0.0700	सरकाराचेर
		सरकाराचो (‘of the government’)	0.0300	
		सरकाराची (‘of the government’)	0.0158	
		सरकाराचे (‘of the government’)	0.0153	

Table 5.22: Example of *K_MLM-BERT* output for Konkani.

The word selected by *K_MLM-BERT* can either be correct or incorrect. Also, sometimes, the system may not provide suggestions; the word remains unchanged in such instances. Examples for each of these cases are discussed in Section 5.6.6 The correct words are indicated in green, and the incorrect words are underlined and marked in red.

5.6.4.3 Error correction using Peter Norvig spell corrector for Konkani

This approach attempts to select the most likely spelling suggestion for the word w . Thus, looking for the correction c that has the highest likelihood of being the intended correction out of all potential candidate corrections suggested by Peter Norvig spell corrector². Here, the corpus provided is the Konkani training corpus as described in Section 5.21. Peter Norvig spell corrector has been named as *Norvig_SC* for convenience.

$$\operatorname{argmax}_{c \in \text{candidates}} P(c | w) \quad (5.1)$$

²<http://norvig.com/spell-correct.html>

By Bayes' theorem equation 5.1 is equivalent to:

$$\operatorname{argmax}_{c \in \text{candidates}} P(c)P(w | c)/P(w) \quad (5.2)$$

Since $P(w)$ is the same for every possible candidate c , it can factor equation 5.2 out, giving equation 5.3:

$$\operatorname{argmax}_{c \in \text{candidates}} P(c)P(w | c) \quad (5.3)$$

5.6.4.4 Error correction by proposed MB-NSC hybrid approach

Automatic error correction ensures the accuracy and reliability of data, saving valuable time and resources. Thus, both the above approaches are combined to suggest a more reliable word. This proposed hybrid approach is named as MB-NSC approach. The algorithm for the MB-NSC is provided below:

Input: OCR_word , $SW_{K_MLM-BERT}$, SW_{Norvig_SC} , $P(SW_{Norvig_SC})$

Output: $out\ put \leftarrow Suggested_word_{MB-NSC}$

```

if  $P(SW_{Norvig\_SC}) == 0$  then
    if  $(Len(SW_{K\_MLM-BERT}) \geq Len(OCR\_word) * 2 \pm 1)$  OR  $(Len(SW_{K\_MLM-BERT}) \leq Len(OCR\_word) / 2 \pm 1)$  then
         $out\ put \leftarrow OCR\_word$  ;
    else
         $out\ put \leftarrow SW_{K\_MLM-BERT}$ ;
    end
else
     $out\ put \leftarrow SW_{Norvig\_SC}$  ;
end

```

Algorithm 4: MB-NSC error correction approach

The MB-NSC approach takes four inputs: i) the OCR-recognized (OCRed) word referred to as OCR_word , ii) the word suggested by the $K_MLM-BERT$ referred to as $SW_{K_MLM-BERT}$, iii) the word suggested through $Norvig_SC$ for Konkani referred to as SW_{Norvig_SC} , and iv) the probability of the suggested word SW_{Norvig_SC} appearing in the Konkani corpus, referred

to as $P(SW_{Norvig_SC})$. The output of this algorithm is the resultant word suggestion named as $Suggested_word_{MB-NSC}$. The system performs the following checks to determine the final output:

1. If the probability of the word suggested by $Norvig_SC$ is zero and the length of the word suggested by $K_MLM-BERT$ model ($Len(SW_{K_MLM-BERT})$) is either half the length of OCR_Word or double the length with a difference of one, then the algorithm selects OCR_Word as the final output. Otherwise, it selects $SW_{K_MLM-BERT}$ as the final output.
2. If the probability of the word suggested by $Norvig_SC$ is not equal to zero, then the algorithm selects SW_{Norvig_SC} as the final output.

The $K_MLM-BERT$ gives contextual appropriate word suggestions. However, the word suggested by $K_MLM-BERT$ may not always be correct. Thus, it check the length of the suggested word. It should not be excessively long or too short. This is based on the observation that Tesseract does the insertion and deletion of 1 or 2 characters per word on average. Further $Norvig_SC$ will suggest the nearest correct word if its probability is not equal to zero. That means the suggested word by $Norvig_SC$ is correct, as it is occurring in the corpus. This helps to maintain or increase the word and character accuracy.

5.6.5 Results and Analysis: Performance of Tesseract Hindi OCR on Konkani documents

The result of Tesseract Hindi OCR performance on *Konkani_OCR_large_test_dataset*, is outlined here. It present the performance and detailed error analysis of Tesseract Hindi OCR on Konkani documents. Evaluation metric used are word accuracy (WA), character accuracy (CA), word error rate (WER) and character error rate (CER). Section 5.6.5.1 details the WER and CER of Tesseract Hindi OCR on *Konkani_OCR_large_test_dataset*. Section 5.6.5.2 details the op 90 characters/ligatures/syllables that caused errors. Section 5.6.5.3 discusses the top 50 words that were incorrectly recognized and words that consistently showed zero accuracy.

5.6.5.1 Performance of Tesseract Hindi OCR on *Konkani_OCR_large_test_dataset*

The evaluation was conducted on the Tesseract Hindi OCR³ using the *Konkani_OCR_large_test_dataset* for testing purposes. The results showed a word accuracy of 89.25% and a character accuracy of 95.85%. The accuracy was evaluated using the ISRI analytic tools for OCR evaluation⁴. In Table 5.23, the Word Error Rate (WER) was found to be 10.75, while the Character Error Rate (CER) was 4.15. The CER was further broken down into insertion errors (1.02%), substitution errors (2.09%), and deletion errors (1.04%). It is worth noting that substitution errors occurred more frequently than insertion and deletion errors.

Dataset	WER	CER			
		Insertion	Substitution	Deletion	Total
<i>Konkani_OCR_large_test_dataset</i>	10.75	1.02	2.09	1.04	4.15

Table 5.23: Word Error Rate (WER) and Character Error Rate (CER) of Tesseract Hindi OCR on *Konkani_OCR_large_test_dataset*

5.6.5.2 Top 90 characters/ligatures/syllables that caused errors

In Table B.1 (refer Appendix B), the top 90 syllables that caused errors during OCR recognition are listed, comprising eleven columns: (i) Serial Number, (ii) Actual Syllable, (iii) Description of Actual Syllable, (iv) Error Frequency, (v) Percentage of Error, and last six columns are the frequencies of the six highest recognized OCR syllables enclosed in parentheses.

5.6.5.2.1 Top most error caused by consonant ष: The character ष ([ə]) accounted for 1.24% of the errors, making it the highest occurring error in OCR output, possibly due to its rare occurrence in the Hindi text. LDC-IL Konkani corpus contains 162661 (4.49%) words that include the character ष ([ə]) having 25070 distinct words. Although consonant ष ([ə]) is used in other languages such as Marathi, Gujarati, Tamil, and Tulu, its usage is limited in Hindi. As a result, the Tesseract Hindi OCR incorrectly recognizes words containing the character ष ([ə]).

³<https://github.com/tesseract-ocr/>

⁴<https://github.com/eddieantonio/ocreval>

5.6.5.2.2 Top ten errors caused by various Devanagari characters : The most frequent errors after that are caused by character ँ (*Anusvara*) and ॅ *candra* [ɛ], contributing to 0.279% and 0.273% of errors respectively. It was observed that the Anusvara character was frequently substituted with a space character, and in some instances, was replaced by vowel sign ँ (U+093A) and candrabindu ॅ as shown in Table B.1 (refer Appendix B). Space character has been indicated by text "<SPC>". The occurrence ॅ *candra* [ɛ] were mostly generated from *Words with multiple modifiers* category, examples हॅलो ([ɦɛlɔ] 'hello'), फॅमिली ([fɛmili] 'family'), टेक्नोलॉजीची ([tɛknolɔdzitsi] 'of technology'). Although these words are not among the most frequent words, they account for 0.88 % of the total word occurrences in both the *candra* and *Words with multiple modifiers* categories within the corpus.

It is also observed that the Right (') and Left (‘) single quotation marks with Unicode values [U+2019] and [U+2018], respectively, introduce errors of around 0.116% and 0.111%, respectively. The fifth most frequent error was related to the character र ([rə]). The punctuation dot (.) is the sixth most frequent error at 0.06%. The character *matra* े, accounting for 0.054% of the errors, is the seventh most frequent error. Thus based on statistics, it can be inferred that the majority of errors primarily originated from characters/syllables such as ँ ([ɛ]), *anusvara*, *candra* ([ɛ]), right, left single quotation mark, punctuation marks (.), and *matras*. Another syllable that contributed to errors was व ([və]), which was often recognized as ब ([bə]), with a frequency of 5316, followed by ल ([lə]), क ([kə]), त ([tə]), <SPC> (Space character), र ([rə]), and उ ([u]). (refer to Appendix B; Table B.1 for further details on the remaining frequent errors).

5.6.5.2.3 Errors caused by double quotation mark, *Nukta* and numbers : Additionally, Right double quotation mark (") also resulted in errors, often recognized as characters such as " (two Right or left single quotation mark) or "" (one right or left double quotation mark and one right or left single quotation mark (see Appendix B; Table B.1 Sr. No 11). Furthermore, it is observed an error for the left double quotation mark (‘) (see Appendix B; Table B.1 Sr. No 31) which produced interesting recognition as the Devanagari syllables, i.e. >*, ', <SPC>, ** and >ब ([bə]). One reason for this is that the character (") appeared adjacent to the Devanagari character without any space, indicating a potential problem with segmentation.

Furthermore, issues were observed with character ः *Nukta* (Consortium, 2022)(see Appendix B; Table B.1 Sr.No 45), question mark (?) (see Appendix B; Table B.1 Sr.No 47), and numerical

digits like 3 and 7, which were incorrectly identified as Devanagari syllables. For instance, the number 7 produced random suggestions like ७, ऋ ([prə]), १, २ (refer to Appendix B; Table B.1 Sr. no 89).

5.6.5.2.4 Errors caused by Devanagari consonants: It was also observed that the Devanagari consonant such as न ([nə]), य ([jə]), ह ([ɦə]), ल ([lə]), ट ([tə]), प ([pə]), ड ([də]), त ([tə]), ठ ([tʰə]), स ([sə]), क ([kə]), र ([rə]), ज ([dʒə]), श ([ʃə]), ण ([ɳə]), म ([mə]), ब ([bə]), द ([də]), ध ([dʰə]), च ([tʃə]), ण ([ra]), ढ ([dʰə]), ण ([gə]), थ ([tʰə]), घ ([gʰə]), भ ([bʰə]), उ ([u]), ष ([ʃə]) (Consortium, 2022) were incorrectly recognized as other consonant, some of which were visually similar while others were not. For example, ल ([lə]) was recognized as न ([nə]), ठ ([tʰə]) was recognized as ट ([tə]), and ड ([də]) was recognized as ह ([ɦə]).

5.6.5.2.5 Errors caused by Independent and Dependent Vowels: The vowel, such as अ ([ə]), were found to have multiple variations when recognized, appearing quite different from one another. For instance, the vowel अ ([ə]) was identified as आ ([a]), थ ([tʰə]), थर ([tʰər]), थआ ([tʰəa]). Additionally, errors were also noticed in the dependent Vowels involving vowel sign ू ([u:]) and ा ([a]) having error percentages 0.007% and 0.005% respectively (see Appendix B; Table B.1 Sr.No 22 and 24). Each of them, in turn, contributes approximately 0.006% of the overall errors.

Errors caused by ligatures and syllables: Some of the instances of actual *ligatures* and syllables that contributed towards errors are listed below, and others are given in Table B.1 (refer Appendix B).

1. द्य ([dʲə]) : Conjunct of consonant letter द ([də]) with *halant*⁵ ् and consonant य ([jə]).
2. त्र ([tʳə]) : Conjunct of consonant letter त ([tə]) with *halant* ् and consonant र ([rə]).
3. त्र ([tʳə]) : Conjunct of consonant letter त ([tə]) with a *halant* ् and consonant letter त ([tə]).
4. ण ([gʳə]) : Conjunct of consonant letter ण ([gə]) with a *halant* ् and consonant र ([rə]).

⁵Halant/virama is phonological concept in Devanagari which explicitly written after consonant will suppress vowel sound at the end of consonant which otherwise default for consonant

5. ङी ([ĩ:]): Consonant ङ ([ə]) with the vowel signs ङी ([i:]) and *anusvara* (◌ं).

6. ङँ ([ɛ]): Consonant ङ ([ə]) with the vowel *candra* ([ɛ]) signँ.

Although these syllables which form conjunct ligature⁶ are similar to Devanagari letter/syllable, such as ङ ([t̪ə]), ङ ([d̪ə]), on recognition, it is important to write them correctly as a combination of Devanagari characters to generate the equivalent syllable. As an example, the conjunct ligature ङ ([t̪rə]) consist of consonant ङ ([t̪ə]) with *halant* ् and consonant र ([rə]). However, the OCR output for these types of conjunct ligatures often differs significantly from the actual conjunct ligature form (see Appendix B; Table B.1, Sr. No 37, 40, 43, 50, 55, 57, 61, 68, 73, 74, 78, 79, 80, 81 and 85). This indicates the various *ligatures* and syllable (consonant with vowel diacritic) contributes towards errors.

5.6.5.3 Top 50 words that were incorrectly recognized and words that consistently showed zero accuracy:

In addition, Table B.2 (refer Appendix B) presents the top 50 incorrectly recognized words. It lists the actual word, its frequencies, and the corresponding recognition accuracy. It is worth noting that approximately half of the words within the top 50 error-prone words contain the character ङ ([ə]), either as a consonant or part of a conjunct consonant. Intriguingly, these specific words consistently displayed a 0% accuracy rate in terms of correct recognition, indicating that the words with character ङ ([ə]) were never correctly identified in any instances found within the corpus. Further analyzing the error, it was found none of the words having character ङ ([ə]) were correctly recognized.

Further Konkani uses character र and its forms of distinctive conjuncts such as Reph र्य (refer Table 5.24), Rakar ङ (formed from characters क , ् , र) and eyelash (̣⁶) (refer Table 5.24). It is important to note that, in Konkani there exist numerous words with eyelash (̣) like दुसऱ्या ([dusəɾja] 'to other/second'), बऱ्या ([bəɾja] 'good'), तिसऱ्या ([t̪isɾja] 'to third'), खऱ्या ([kʰəɾja] 'truth'), बऱ्याचि ([bəɾjatsə] 'after many'). Thus incorporate the use of Zero-Width Joiner, resulting in visual variations of the same word, as exemplified in Table 5.24. For example word दुसऱ्या ([dusəɾja] 'to other') has characters द ([d̪ə]), ु (U), स ([sə]), र ([rə]), ् (*halant*), [U+200D] Zero-Width Joiner (ZWJ), य ([jə]), ा ([a]) and word दुसऱ्या ([dusəɾja]) has same characters as in

⁶<https://www.unicode.org/versions/Unicode15.1.0/ch12.pdf>

Category	Syllable	Character combination and Unicode			
ZWJ	य	र	्	(Zero Width joiner)	य
		[U+0930]	[U+094D]	[U+200D]	[U+092F]
ZWNJ	य	र	्	य	
		[U+0930]	[U+094D]	[U+092F]	

Table 5.24: An example words having Zero Width Joiner (ZWJ) and Zero Width Non-Joiner (ZWNJ)

previous example except ZWJ. Thus word gets a different appearance for the same set of characters with and without ZWJ, as shown in Table 5.24 and Section 3.5.2. . And it is observed that the word दुसऱ्या is incorrectly recognized for every occurrences, which is a frequent word in Konkani (see Table B.2 entry No 22 (refer Appendix B)). It is observed that words with eyelash are consistently misclassified.

Another example of word प्रयत्न having ZWJ, which is consistently misclassified. This word प्रयत्न ([prəjətnə] 'attempt') as characters प ([pə]), ् (halant), र ([rə]), य ([jə]), त ([tə]), ् (halant), [U+200D](ZWJ), न ([nə]). Here, the word having the same characters without ZWJ appears like प्रयत्. ([prəjətnə] 'attempt') Konkani corpus contains 0.5% words with ZWJ. The words containing ZWJ consistently displayed an accuracy rate of 0%. These words are also among the top 50 erroneous words, as indicated in Appendix B; Table B.2 (entries 22 दुसऱ्या ([dusərja] 'to other/second') and 42 यत् ([jətnə] 'attempt')), both having an accuracy of 0%. Conversely, the same words presented in Appendix B; Table B.2 (entry 27, i.e. यत् ([jətnə] 'attempt') without the ZWJ achieved an accuracy of 7.0%. Therefore, the presence of the words recognizing ZWJ poses a challenge in accurately recognizing the syllables, ultimately diminishing word accuracy. It was also observed that most of the words having errors belonged to *Words with multiple modifiers* category. This can be observed in Table B.2 (refer Appendix B) all the words with error are having more than one modifier.

It was observed that the most frequent word appearing among the top 10 words, i.e. हेँ ([fiẽ] 'this') was incorrectly recognized as हैँ ([fiəẽ]), which is a valid word in Hindi, and नहीँ ([nfi] 'no') was recognized as नहीँ ([nəfi]), which again valid word in Hindi. Additionally, the word मळार ([mə[ar] 'in the field') was recognized as मार ([mar] 'beat'), which is a valid word in Hindi as well as Konkani, thus indicating instances of real errors, i.e., the recognized word is a valid

word in the same language.

5.6.6 Result and Analysis of MB-NSC on Konkani data

Section 5.6.6.1 details the Performance of MB-NSC error correction on *Konkani_EC_test_dataset*.

Section 5.6.6.2 error analysis of MB-NSC error correction approach.

5.6.6.1 Performance of MB-NSC on *Konkani_EC_test_dataset*

Accuracy	Tesseract Hindi OCR	<i>K_MLM-BERT</i>	<i>Norvig_SC</i>	MB-NSC	Improvement
	(A)	(B)	(C)	(D)	E=D-A
WA	89.11%	91.80%	89.19%	91.85%	2.74%
CA	95.15%	94.09%	96.16%	97.75%	2.60%

Table 5.25: Word accuracy (WA) and Character accuracy (CA) of *K_MLM-BERT* and *Norvig_SC* and MB-NSC approach on *Konkani_EC_test_dataset*.

The evaluation of error correction approaches *K_MLM-BERT*, *Norvig_SC* and a MB-NSC approach is conducted on *Konkani_EC_test_dataset* using the ISRI analytic tool for OCR evaluation⁷. The result revealed that the existing Tesseract Hindi OCR showed a word accuracy of 89.11% and character accuracy of 95.15% on the *Konkani_OCR_subset_test_dataset*. Subsequently, after applying the error correction using the proposed MB-NSC approach, an enhancement in word accuracy was observed, reaching 91.85% and character accuracy was 97.75% on *Konkani_EC_test_dataset*. This word accuracy achieved by the MB-NSC approach is similar to the word accuracy of existing Hindi OCR, i.e., 91.60% as reported (Mathew et al., 2016).

This performance of MB-NSC approach surpasses the performance of other proposed error correction approaches. Specifically, the *K_MLM-BERT* method demonstrated a word accuracy of 91.80% and character accuracy of 94.09%. Here, character accuracy of *K_MLM-BERT* decreased due to contextual word suggestion, as the word length of the suggested word may not be equal to the OCR'd word. Whereas the *Norvig_SC* approach achieved word accuracy of 89.19% and character accuracy of 96.16% on *Konkani_EC_test_dataset*. These results are presented in Table 5.25.

⁷<https://github.com/eddieantonio/ocreval>

The *Norvig_SC* is particularly adept at successfully correcting words with one or two characters, especially vowels and consonants, that have been incorrectly recognized in the word. It lacks the capability to offer accurate suggestions due to incorrect recognition of conjunct consonant or modifies. And the *K_MLM-BERT* model suggests words based on context. Thus, the proposed MB-NSC approach yielded a word accuracy of 91.85% and character accuracy of 97.75%. Overall, there was a 2.74% increase in word accuracy and a 2.60% increase in character accuracy (see Table 5.25). Additionally, Table 5.26 shows that insertion and substitution errors were

(OCR /post processing) /Types of errors	Insertion	Substitution	Deletion	CER
Tesseract Hindi OCR	1.37	2.29	1.19	4.85
MB-NSC approach	0.47	0.98	0.80	2.25

Table 5.26: Insertion, substitution, deletion error rate and Character Error Rate (CER) on *Konkani_EC_test_dataset*

reduced by applying the MB-NSC approach. Moreover, deletion errors were partially reduced.

Word category	Tesseract Hindi OCR WER %	MB-NSC approach WER %
<i>Simple syllable (Without modifiers)</i>	1.20	0.23
<i>Matras</i>	1.47	1.09
<i>Eekar</i>	0.55	0.38
<i>Ookars</i>	0.33	0.25
<i>Conjunct consonant</i>	2.06	1.79
<i>Anusvara</i>	0.98	0.43
<i>Candra</i>	0.08	0.05
<i>Words with multiple modifiers</i>	4.22	3.93
Total WER	10.89	8.15

Table 5.27: Word error rate (WER) of Tesseract Hindi OCR and MB-NSC approach for each category of words

Table 5.27 shows the category-wise WER for the output of the Tesseract Hindi OCR and MB-NSC approach. It is observed that most of the errors are introduced by the categories (a) *Words*

with multiple modifiers, Conjunct consonants, (b) Matras and (c) Simple syllable (Without modifiers). This is mainly due to conjunct consonants and character ँ and at some extend due to misclassification of matra such as े (U+0948) to ै (U+0947)). Further it is observed that these errors are reduced using MB-NSC approach. It is also observed that there is marginal improvement in categories like *Eekar*, *Ookar* and *Candra*. It is important to understand that these improvement are due to contextual information and the likelihood of finding the most probable word within the corpus.

5.6.6.2 Error analysis of MB-NSC approach

Here the section 5.6.6.2.1 shows output given by MB-NSC approach. Section 5.6.6.2.2 shows the use cases that were used for testing MB-NSC approach. Section 5.6.6.2.3 provides examples of words corrected by proposed MB-NSC approach having character ँ (i.e. Specific to Konkani). It also provides examples of words that had additional character *Anusvara* that are corrected by MB-NSC approach. Section 5.6.6.2.4 provides the examples of words that are not corrected by MB-NSC approach.

5.6.6.2.1 Output of MB-NSC approach The findings of this work suggest that the *K_MLM-BERT* provides more precise recommendations (see Table 5.28). For example, the actual word कार्यावळीचो ([karjavəli:tsə] 'of the program') is recognized as कार्यावळीचो ([karjavəki:tsə]) by Tesseract OCR whereas the proposed approach corrects it to कार्यावळीचो ([karjavəli:tsə] 'of the program'). Let's consider another example to illustrate the effectiveness of the proposed approach: The actual word परीकारान ([pəri:karān]) is recognized by OCR as पररीकारान ([pərəri:karān]) while the correction provided by the suggested model is परीकारान ([pəri:karān]) (see Table 5.28 for more instances). It is important to note that for each example in Table 5.28, *Norvig_SC* provided a suggestion same as the OCR'd word having 0 probability, which indicates this suggested word never occurred in the Konkani corpus (Ramamoorthy et al., 2019). Thus, the output of *K_MLM-BERT* was considered as final based on MB-NSC approach.

Further, the Table 5.29 shows instances of the word suggested by *K_MLM-BERT*, *Norvig_SC* and MB-NSC approach. For example, the actual word पळयतकच ([pəʃəʃəʃəkəts] 'after seeing') was recognized as पढयतकच ([pəḍʰəʃəʃəkəts]), the *K_MLM-BERT* suggested word as हाव ([ɦav] 'me') and suggestion provided by *Norvig_SC* was पळयतकच ([pəʃəʃəʃəkəts]). Thus, the

Sr. No.	Actual word	OCRed output	<i>K_MLM-BERT</i> output
1	जाळवणदार ('networker')	जारवणदार	जाळवणदार
2	जाळवणदार ('networker')	जाछवणदार	जाळवणदार
3	पर्रीकार ('Parrikar' (last name))	पररीकार	पर्रीकार
4	पर्रीकारान ('by Parrikar')	पररीकारान	पर्रीकारान
5	हळदणकार (Haldankar (last name))	हछदणकार	हळदणकार
6	अधिकार्याचो ('of the authorities')	अधिकार्याचो	अधिकार्याचो
7	अधिकार्याक ('to the authorities')	अधिकार्याक	अधिकार्याक
8	अधिकार्यानी ('the authorities')	अधिकार्यानी	अधिकार्यानी
9	विशींची ('about the')	विशींची	विशींची
10	कार्यावळीचो ('of the program')	कार्यावकीचो	कार्यावळीचो
11	मॅडिकल ('medical' (foreign word))	मॅडिकल	मॅडिकल
12	वर्साचो ('of the year')	वर्साचो	वर्साचो
13	फायदो ('advantage')	फायदो	फायदो
14	फॅडररान ('by (Roger) Federer')	फॅडररान	फॅडररान
15	मळार ('in the field')	मार	मळार
16	सकयल्या ('lower')	सकयलया	सकयल्या

Table 5.28: Output of *K_MLM-BERT* indicating suitable output as per context.

final output based on the MB-NSC approach, i.e. पळयतकच ([pəʃəjətəkəts]), which is the correct suggestion. It is important to note that the *Norvig_SC* suggestion turns out to be consistent. For example the actual word वळख ([vəʃəkʰ] 'identity'), which is recognized as वठख ([vəʃəkʰ]), and *K_MLM-BERT* output as चवकशी ([tʃəvkəʃi:] 'investigation') and *Norvig_SC* provided output as वळख ([vəʃəkʰ] 'identity') with $P(\text{वळख}) > 0$. Thus, selecting वळख ([vəʃəkʰ]) word as the final output based on the MB-NSC approach. It is important to note that the actual word वळख ([vəʃəkʰ] 'identity') appeared in the top fifty words, contributing to the error (refer Appendix B; Table B.2).

5.6.6.2.2 Use cases used for testing MB-NSC approach Additionally, Table 5.30 shows examples of five generic possible cases of contextual errors. These cases are as per the proximity

Sr. No.	Actual word	OCRed word	<i>K_MLM-BERT</i> output	<i>Norvig_SC</i> output	MB-NSC approach output
1	कळटकच (after knowing)	कछटकच	no suggestion	कळटकच	कळटकच
2	जनमत (mandate)	जनममत	no suggestion	जनमत	जनमत
3	पळयतकच (after seeing)	पढठयतकच	हाव	पळयतकच	पळयतकच
4	वळख (identity)	वछख	एक	वळख	वळख
5	वळख (identity)	वठख	चवकशी	वळख	वळख
6	नितळसाण (cleanliness)	नितड्साण	नितळसाण	नितड्साण	नितळसाण
7	धन्यताय (satisfy)	धन््यताय	धन्यताय	धन््यताय	धन्यताय
8	पर्रीकार (Parrikar (last name))	पररीकार	पर्रीकार	पररीकार	पर्रीकार
9	उमळशीक (Washing)	उमछशीक	उमळशीक	उमछशीक	उमळशीक
10	ल्हानपणा (smallness)	ल्ल्हानपणा	ल्हानपणा	ल्ल्हानपणा	ल्हानपणा

Table 5.29: Output of *K_MLM-BERT*, *Norvig_SC* and MB-NSC approach

of incorrect words in the sentence. Apart from OCRed text, the example in Table 5.30 also provides its transliteration, English translation, actual text, and MB-NSC output. The five cases are as follows:

1. Case 1: Only one word is incorrect in a sentence.
2. Case 2: Non-consecutive two words incorrect in a sentence.
3. Case 3: Consecutive two words incorrect in a sentence.
4. Case 4: Majority of words incorrect in a sentence.
5. Case 5: Number or quantity words incorrect in a sentence.

The proposed MB-NSC approach exhibits contextual-based corrections, as evidenced by the examples in Table 5.30 case 1 and case 2. For example the actual text मळार ([məlar] 'in the field') was incorrectly recognized as मार ([mar] 'beat') and the MB-NSC approach suggests मळार ([məlar], 'in the field') (see Table 5.30 case 2). It is important to note that the word मळार ([məlar] 'in the field') appeared among the most frequent words contributing towards error (see Appendix B; Table B.2). Case 3 shows an example of two words consecutively incorrect. However, the MB-NSC approach corrects words having one or two character misclassifications. Case

Case 1: Only one word incorrect in a sentence.	
Actual text	साडेसंक सुरु जातली
Transliteration	sadesaMka suru jatall
English translation	Will start at 6.30
OCRed Text	साडेसंक सुरु <u>जातली</u>
MB-NSC output	साडेसंक सुरु जातली
Case 2: Non consecutive two words incorrect in a sentence.	
Actual text	सचिनान आपलो 49 वो शेकडो नोंदयल्या उपरांत आंतरराष्ट्रीय मळार 100 परसय चड शेंकड्यांची नोंद जाल्या.
Transliteration	sachInAn Apalo 49 vo shekado noMdayalyA uparAMta AMtararAstrIya maLAra 100 parasaya chada sheMkadyAMchI noMda jAlyA.
English translation	Sachin (Tendulkar) has recorded more than 100 international centuries after his 49 th (ODI) century.
OCRed text	सचिनान आपलो 49 वो शेकडो नोंदयल्या उपरांत आंतरराष्ट्रीय <u>मार</u> 100 परसय चड <u>शेंकडांची</u> नोंद जाल्या.
MB-NSC output	सचिनान आपलो 49 वो शेकडो नोंदयल्या उपरांत आंतरराष्ट्रीय मळार 100 परसय चड <u>शेंकडांची</u> नोंद जाल्या.
Case 3: Consecutive two words incorrect in a sentence.	
Actual text	खोलायेन तपास करुन तिची वळख सोदून काडिल्ली
Transliteration	kholAyena tapAsa karuna tichI vaLakha sodUna kadillI
English translation	Her identity had been discovered after a thorough investigation
OCRed Text	खोलायेन तपास करुन तिची <u>वठख</u> <u>सोदून</u> काडिल्ली
MB-NSC	खोलायेन तपास करुन तिची वळख <u>सोदून</u> काडिल्ली
Case 4: Majority of words incorrect in a sentence.	
Actual text	अधिकारी कुमारस्वामी हांचेर टिका करतना
Transliteration	adhikArI kumArasvAmI haMchera tika karatAnA
English translation	while criticizing Officer Kumarswami
OCRed text	अधिकारी <u>कुमाररब्बेवामी</u> <u>डाके</u> टिका <u>करतना</u>
MB-NSC output	अधिकारी <u>कुमाररब्बेवामी</u> <u>डाके</u> टिका करतना
Case 5: Number or quantity words incorrect in a sentence	
Actual text	णव मेरेन महेंद्र गांवकार आनी कलाकार आसले
Transliteration	Nava merena maheMdra gAMvakAra AnI kalAkAra Asale
English translation	Mahendra Gaonkar and artists were there till nine
OCRed text	<u>गाब</u> मेरेन महेंद्र <u>मांबकार</u> आनी कलाकार आसले
MB-NSC output	<u>आता</u> मेरेन महेंद्र <u>मांबकार</u> आनी कलाकार आसले

Table 5.30: Use cases tested with proposed MB-NSC approach

4 shows the example of the majority of words being incorrect but not continuous. In this case, MB-NSC provides contextual suggestions. Case 5 shows an example where the model cannot handle appropriate suggestions for numbers. The MB-NSC approach provides contextual words; however, it may not be a number.

Actual word	OCRed out-put	MB-NSC out-put	Frequency of actual word in Konkani corpus
सगळे (all)	<u>सगळे</u>	सगळे	1887
मळार ('in the field')	<u>मार</u>	मळार	812
वळख (identity)	<u>वळख</u>	वळख	736
पळयत (looking at it)	<u>पळयत</u>	पळयत	669
पळोवपाक (to see)	<u>पळोवपाक</u>	पळोवपाक	614
सगळी (all)	<u>सगळी</u>	सगळी	595
पळयतात (seeing)	<u>पळयतात</u>	पळयतात	265
मेळप (meeting)	<u>मेळप</u>	मेळप	214
मेळपी (meet)	<u>मेळपी</u>	मेळपी	186
उमळशीक (exciting)	<u>उमळशीक</u>	उमळशीक	186
आडखळ (hurdle)	<u>आडखळ</u>	आडखळ	177

Table 5.31: Words corrected by proposed MB-NSC approach having character ळ, along with its frequency occurring in Konkani corpus

Actual word	OCRed word	MB-NSC output
खबरो (news)	<u>खबरो</u>	खबरो
एकसुरो (monotonous)	<u>एकसुरो</u>	एकसुरो
घेवपाचो (to be taken)	<u>घेवपाचो</u>	घेवपाचो
ओवरींनी (with overs)	<u>ओवरींनी</u>	ओवरींनी
संदीचो (of the opportunity)	<u>संदीचो</u>	संदीचो
सैमाचो (of nature)	<u>सैमाचो</u>	सैमाचो
गोंयचो (Goa's)	<u>गोंयचो</u>	गोंयचो
घेवपाचो (to be taken)	<u>घेवपाचो</u>	घेवपाचो

Table 5.32: Examples of words having additional *Anusvara* that are corrected by MB-NSC approach

5.6.6.2.3 Examples of words corrected by MB-NSC approach having character ळ and *Anusvara*: Table 5.31 illustrates the actual word with character ळ ([ə]), the OCR-recognized output, the corrected word by the proposed MB-NSC approach, and its frequency of appearance

in the gold standard Konkani corpus (Ramamoorthy et al., 2019). As a result, the proposed system effectively corrects the frequently occurring words with the character ँ ([ə]). Table 5.32 shows the words showing action specific to the Konkani language were correctly suggested by the MB-NSC approach. Overall, upon performing character analysis, it was observed that the MB-NSC approach significantly corrected words with characters ँ ([ə]), which gave zero accuracy for the words having this character before error correction was applied. Also the errors contributed by character ँ (anusvara) and ॅ ([e])(candra) were reduced by MB-NSC approach. It is important to note that these characters were appearing among the top 3 errors.

Sr. No.	Actual word	OCRed output	<i>K_MLM-BERT</i> output	<i>Norvig_SC</i> output	MB-NSC output
1	वेगवेगळ्या (various)	वेगवेगळ्या	दोन	वेगवेगळ्या	वेगवेगळ्या
2	हणजुणें (Ajuna (place))	हणजुणें	अवतार	हणजुणें	हणजुणें
3	सॅबॅस्तियन (Sabastian (Name))	सॅबॅस्तियन	पण	सॅबॅस्तियन	सॅबॅस्तियन
4	कॅसेटी (cassettes)	कॅसेटी	no suggestion	कॅसेटी	कॅसेटी
5	फ्रॅण्ड्स (friends)	फ्रॅण्ड्स	no suggestion	फ्रॅण्ड्स	फ्रॅण्ड्स

Table 5.33: Incorrect word suggested by *Konkani_Error_Correction_MLM-BERT*(*K_MLM-BERT*) and *Norvig_SC*

5.6.6.2.4 Example of words not corrected by MB-NSC approach The Table 5.33 shows instances of incorrect suggestion provided by both *K_MLM-BERT* and *Norvig_SC* and MB-NSC approach. For example the actual word वेगवेगळ्या ([vegəvegʌja] 'various') that was recognized as वेगवेगळ्या ([vegəvegʌja]), where *K_MLM-BERT* provided output as दोन ([don] 'two') and *Norvig_SC* provided output as वेगवेगळ्या ([vegəvegʌja]) having probability 0 based on algorithm 4. Both models do not provide correct suggestions for a less frequent word in the corpus, as shown in Table 5.33 Sr. No 5 and 6. E.g., actual word फ्रॅण्ड्स ([frɛɳɖs] 'friends'), OCR recognized word फ्रॅण्ड्स ([frɛɳɖs]) and *K_MLM-BERT* did not product any suggestion and *Norvig_SC* provided suggestion फ्रॅण्ड्स ([frɛɳɖs]), which is same as OCR recognized word. These English words have been transliterated in Devanagari, whose occurrence is significantly less in the corpus, with only 1 or 2 occurrences. This model basically cannot give correct suggestions for quantity, e.g. णव ('nine'). It also struggles to suggest accurate foreign English words if their frequency is less in the training data. It could not provide suggestion for words with ZWJ.

5.7 Conclusion

Initially performance of the existing OCRs for two different fonts *Aparajita* and *Nakula* was studied. It was observed that OCR accuracy differs for different fonts. Moreover, no multi font OCR was available. Accordingly, Weighted Syllable based Spell Correction Approach (WSSC) was proposed for suggesting the most appropriate word based on the error analysis for two fonts *Nakula* and *Aparajita*. Weighted Syllable based Spell Correction Approach (WSSC) was tested for all three OCR outputs (Tesseract, Indsenz and e-Aksharayan) with *BM_Aparajita* dataset and *BM_Nakula* dataset. The accuracy of Tesseract OCR output for *BM_Aparajita* dataset increased from 65.95% to 67.19%. Similarly, the Indsenz accuracy on *BM_Aparajita* dataset increased from 53.23% to 55.00%. Small marginal increase was also observed in the OCRs accuracy for *BM_Nakula* dataset. To handle new errors, the frequency-based error mapping tuple needs to be updated, thereby demonstrating its robustness. It handles substitution errors but fails to handle insertion and deletion errors. Another challenge faced during post-processing is due to the inflectional nature of Indian languages. Thus, motivation to provide suggestions for the incorrect word based on context. Hence, another error correction approach using MLM-BERT was proposed. It has demonstrated its effectiveness for Hindi with a substantial improvement of 3.58%, showing 95.18% word accuracy, when evaluated on a Hindi OCR dataset. This approach performs well by providing context-sensitive suggestions. Nevertheless, the incorrect suggestions often align closely with the actual word or most likely based on context. It was also observed that in some cases, the model does not provide any suggestions. These discrepancies may arise due to incorrect consecutive words or contextual considerations. The suggested word length could be much greater or quite less than the OCR'd word. This affects the character accuracy. Moreover, to achieve the main objective of this research of proposed OCR solution for Konkani documents. Use of Tesseract Hindi OCR on Konkani documents was explored, yielding a word accuracy of 89.25%. The evaluation showed a CER of 4.15% comprising insertion errors (1.02%), substitution errors (2.09%), and deletion errors (1.04%). The topmost error was primarily associated with the syllable $\overline{\text{a}}$ ([a]). Therefore, the utilization of Hindi OCR for Konkani text has presented specific challenges in terms of performance. This lays down ground work for significantly contributing to developing and subsequently improving the accuracy of Konkani OCR'd text. This is achieved through the proposed MB-NSC error correction approach, i.e., combining the state-of-the-art Masked Language Modeling BERT and Peter Norvig spell corrector for Konkani. The

MB-NSC approach for Konkani leveraged contextual information to provide precise word corrections, effectively handling words with special characters unique to the Konkani language, including those containing the character $\overline{\text{om}}$ ([om]). Thus accomplishing a 2.74% enhancement in word accuracy and 2.60% of character accuracy compared to the current Tesseract Hindi OCR and demonstrating its efficacy in rectifying errors in the OCR output text during post-processing.

This page is intentionally left blank.

Chapter 6

Algorithm Proposed and Implemented for Error Detection: Devanagari Multilingual Model

6.1 Background and Motivation

Post-processing has two parts, namely, error detection and error correction. Error detection detects errors, and error correction corrects the same. On detecting the OCR errors, only those words can be corrected using error correction techniques to improve the overall accuracy. Hence, error detection becomes an equally important phase in post-processing. In chapter 5, error detection was performed using a dictionary-based approach separately for Konkani and Hindi. Work was reported on error detection for Indian languages such as Hindi and Sanskrit, along with Malayalam, Kannada, and Gujarati. Most existing research on OCR error detection for Devanagari has primarily focused on the Hindi language, achieving an F-score of 0.82 using RNN (Vinita and Jawahar, 2016) (refer chapter 2; Section 2.3.1). However, Devanagari is used by ten languages. It would be beneficial to have a single error detection model in Devanagari for all Indian languages. Currently, no work has been reported to identify post-OCR error for Devanagari in a multilingual document. However, challenges involved in detecting OCR errors for Indian languages are their script complexity, their inflectional nature as indicated by (Sankaran and Jawahar, 2013), poor availability of language resources, and real errors concerning context, i.e., a wrongly recognized word might be valid in the same language. For example, the Konkani valid word मोगाळ [mɔgaɭ] is incorrectly recognized as मोगा [mɔga], which is a valid word in the

Konkani language (refer Table 6.1). Here, the incorrectly recognized words are indicated in red and underlined. The correctly recognized words are in green. Another concern is that a word incorrectly recognized in one language may be valid in another, as shown in Table 6.1. For example, the actual Konkani word हें [hẽ] is incorrectly recognized by OCR, giving output as हैं [həẽ], which is a valid word in Hindi. In such cases, a language-based error detection model will aid in correctly identifying errors, unlike other traditional methods such as the dictionary-based or n-gram error detection approaches.

Example 1: Konkani word incorrectly recognized is valid word in Konkani language	
Actual text (Konkani)	म्हाका मोगाळ आनी जतनाय घेवपी सांगाती जाय.
English translation	I need loving and caring life partner.
OCRed output	म्हाका <u>मोगा</u> आनी जतनाय घेवपी सांगाती जाय.
Example 2: Konkani word incorrectly recognized is valid word in Hindi language	
Actual text (Konkani)	हें सगळें आयकुपाची तयारी दवरुंक जाय.
English translation	You have to be prepared to hear all this.
OCRed output	<u>हें सगळें</u> आयकुपाची तयारी दवरुंक जाय.

Table 6.1: Actual text in Konkani and its OCR recognized output

This work creates a multilingual model for OCR error detection in Devanagari OCR using mBERT with CNN which supports four languages namely Hindi, Konkani, Marathi, and Nepali. The following were the challenges involved in creating a multilingual Devanagari model specific to one script:

- Limited resource languages:** It is crucial to have an OCR system specifically designed for the target language to detect OCR errors. However, in the case of languages like Konkani, an OCR system needs to be made available.
- Inter-language misclassification:** The OCR models may incorrectly classify words that belong to one language as valid. For example, an invalid word in Konkani may be incorrectly classified as valid in Hindi or vice versa. As illustrated in Table 6.1. This indicates the need for improved language differentiation and handling of language-specific characteristics.

- iii. Domain-specific word detection: OCR systems may face difficulties in detecting words from specific domains or specialized vocabularies. For instance, English words transliterated into Indian languages might not be correctly recognized or classified as errors.
- iv. Code Mixing Challenges: Sentences or texts that combine multiple languages, such as Konkani and Hindi, pose challenges for OCR error detection. The models may struggle to accurately classify and process code-mixed content, leading to lower accuracy in identifying errors.

This chapter is organized as follows: Section 6.2 provides the structure of the proposed solution to create a multilingual model for Devanagari. Section 6.3 provides details about dataset used for this work. Section 6.4 discusses the performance of existing Hindi OCR on Konkani, Marathi and Nepali. Section 6.5 details the experiment setup for error detection using Multilingual BERT. Section 6.6 presents the result. Section 6.7 concludes the chapter.

6.2 Structure of proposed solution

This work initially studied the performance of existing Hindi OCR on the *Indic-OCR-test-dataset*, which supports four languages: Hindi, Konkani, Marathi, and Nepali. The best OCR'd output for all these four languages was selected. This optimal OCR was then used to generate the OCR'd text to create the *Indic-post-OCR-dataset* (refer Chapter 4; section 4.3.6), which was used for training and testing the multilingual Devanagari model. Next, fine-tuning of the mBERT tokenizer and model was carried out, leading to the creation of the multilingual Devanagari model for OCR error detection. This model utilized mBERT with CNN and the *Indic-post-OCR-dataset*, supporting the four languages: Hindi, Konkani, Marathi, and Nepali. This work aimed to address the low-resource, inter-language classification challenges and present an approach to identify OCR errors not only in Hindi but also in three more languages: Konkani, Marathi, and Nepali.

6.3 Datasets

There are two dataset used for this experiment (i) *Indic-OCR-test-dataset* to test OCR having four languages and (ii) *Indic-post-OCR-dataset* used to train and test error detection approach (refer Chapter 4; section 4.3.6). Table 6.2 shows the characteristics of *Indic-post-OCR-dataset* that has been used for training and testing the proposed model for Devanagari OCR error detection.

Language	Training		Testing	
	Correct words	Error words	Correct words	Error words
Hindi	153162	49054	42797	3888
Konkani	98194	45095	32720	2357
Marathi	151579	64367	44490	4833
Nepali	154050	88310	44534	10014

Table 6.2: Characteristics of *Indic-post-OCR-dataset*

6.4 Performance of Hindi OCR on languages: Konkani, Hindi, Marathi and Nepali

Dataset	Tesseract OCR		Indsenz OCR	
	CA	WA	CA	WA
<i>Hindi-Indic-OCR-test-dataset</i>	95.87%	91.24%	93.30%	84.88%
<i>Konkani-Indic-OCR-test-dataset</i>	95.19%	88.1%	83.55%	50.6%
<i>Marathi-Indic-OCR-test-dataset</i>	96.39%	91.45%	93.13%	79.5%
<i>Nepali-Indic-OCR-test-dataset</i>	95.95%	87.96%	88.44%	62.29%
CA: Character Accuracy & WA: Word Accuracy				

Table 6.3: Performance of Tesseract and Indsenz OCR using *Indic-OCR-test-dataset*

This work has studied the performance of available Hindi Devanagari OCRs, namely, Tesseract and Indsenz, on the test datasets *Hindi-Indic-OCR-test-dataset*, *Konkani-Indic-OCR-test-*

dataset, *Marathi-Indic-OCR-test-dataset*. and *Nepali-Indic-OCR-test-dataset*. This study was carried out to select the OCR that gives higher accuracy for Hindi, Konkani, Marathi, and Nepali. Table 6.3 shows the word and character accuracy of our test datasets on Tesseract and Indsenz OCR of the Hindi language. It is observed that Indsenz OCR word accuracy varies from 50.6% to 84.88%, and Tesseract OCR word accuracy varies from 87.96% to 91.45%. Tesseract OCR performed better for all four languages. Hence, the Tesseract OCR of the Hindi language was selected to perform text recognition for all four languages and further to work on OCR error detection.

6.5 Experimental setup for Error Detection using mBERT

This section describes the procedure used to create a model to detect errors from Devanagari OCR'd text for four languages, Hindi, Konkani, Marathi, and Nepali, using Bidirectional Encoder Representations from Transformers (BERT) with Convolutional Neural Network (CNN). BERT is a multi-layer bidirectional transformer encoder pre-trained on extensive unlabeled data. It performs two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). The BERT models work at the sentence and token level. Also, BERT needs fine-tuning to handle a specific problem. If the BERT model does not support a specific language, it needs a training model with the data. Here, the error detection problem works at the token level, classifying the token as either valid (label 0) or invalid (label 1). Figure 6.1 shows the proposed system. It has five modules as follows:

- I. To create label data.
- II. Training BERT tokenizer.
- III. Training BERT model.
- IV. Preparing input to train and test error detection model.
- V. Building Error Detection model.

Each of these modules is explained in detail below. Figure 6.2 further shows the input and output of modules I and IV in detail with examples.

I. To create label data:

Each file in *Indic-post-OCR-dataset* consists of three parts (a) [OCR_toInput], (b) [OCR_aligned], and (c) [GS_aligned] (For more details refer Chapter 4; Section 4.3.6). The sentences have been extracted from [OCR_aligned] and [GS_aligned] text from the dataset using NLTK English sentence tokenizer. Further, it split these sentences from [OCR_aligned] and [GS_aligned] text into words based on space. Next, it assigns label 0 if the OCR'd word equals words from the GS_aligned sentence. Else, label 1 is assigned. This dataset was split into 65:15:20 ratios for training, validation, and testing for each language.

II. Training BERT tokenizer:

OCRed text
मंच पर एक कमरा जिसमें थोड़ा सा फर्नीचर एक किताबों की रैक
BERT "bert-base-multilingual-cased" tokenizer output
म, ##ं, च, पर, एक, कम, ##रा, जिसमें, थ, ##ोड़, ##ा, स, ##ा, फ, ##र, ##्नी, ##च, ##र, एक, कि, ##ता, ##ब, ##ों, की, र, ##ैक,
BERT Dev-multi-lingual tokenizer output
मंच, पर, एक, कमरा, जिस, ##में, थ, ##ोड़, ##ा, सा, फर, ##नी, ##चर, एक, किताब, ##ों, की, रैक, ,

Table 6.4: Multilingual BERT Tokenizer output

The BERT model requires the input data in a suitable format to obtain the corresponding embedding of the sentence. The input to the BERT model is a sentence of a fixed length. Each token was given a distinct ID when the BERT model was trained. So, in order to employ a pre-trained BERT model, each token in the input sentence must first be converted into its corresponding unique ID. Sentences are padded to achieve a fixed length for input. BERT tokenizer is essential to obtain unique token ID. The BERT tokenizer will split the word into two subwords (token). The first token is frequently seen word (prefix) in a corpus. Two hashes are prefixed to the second token to indicate that it is a suffix following some other subwords. Further, it converts subwords to unique IDs using the BERT tokenizer. The label of subwords is the same as that of the actual word. The *bert-base-multilingual-cased* tokenizer supports 104 languages, which includes Hindi,

Marathi, and Nepali (Devlin et al., 2019). Table 6.4 shows the actual text and output of *bert-base-multilingual-cased* tokenizer. Here, it is observed that a word that doesn't need to be tokenized still goes through the process and creates a token as a character. Example मंच [māṇṇ] word is tokenized as म, and ##ँच, similarly the word किताबों [kiṭābō] is tokenized as कि, ##ता, ##ब, and ##ों. However, to make the tokenizer more suitable for these languages, the *bert-base-multilingual-cased* tokenizer was trained with the language data and termed it as *Dev-multi-lingual*. The output of the *Dev-multi-lingual* can be seen in the third row of Table 6.4. In this case, word मंच is not tokenized further, and the word किताबों is tokenized correctly as किताब and ##ों. Thus, subword splitting gives better performance. The corpus used for training Hindi, Konkani, Marathi, and Nepali is shown in Table 4.9. Additional Konkani corpus was also taken from the Konkani LDC-IL (Ramamoorthy et al., 2019) and "Konkani Viswakosh"¹

III. Training BERT model:

The BERT Model "*bert-base-multilingual-cased*" supports Hindi, Marathi, and Nepali. However, using this model, it was difficult to achieve a satisfactory F-score to classify errors. Because of this, the BERT *Dev-multi-lingual* tokenizer is used and a corpus to train further the pre-trained BERT model "*bert-base-multilingual-cased*". The corpus used here is the same as that used in module II to build a tokenizer.

IV. Preparing input to train and test error detection model :

The BERT *Dev-multi-lingual* tokenizer is built and used to tokenize the words into tokens. Further, the tokens were converted to their IDs, creating an attention mask. Then, the token ID, label, and attention mask were given as input to the next step of the model-building module. This was done for training, validation and testing data.

V. Building Error Detection model:

The BERT Model *Dev-multi-lingual* and CNN trained with the labeled training data created in module IV to classify words in a sentence as error or otherwise. Here, BERT embeddings were given to CNN. The CNN architecture has four convolutional layers with four different kernel sizes, i.e. (2, 3, 4, and 5), having 32 filters. Further, there is a max-pool layer. The final layer is the linear layer, which takes the output from the maxpool

¹https://wikisource.org/wiki/Index:Konkani_Viswakosh_Vol1.pdf

layer. The maxpool layer and convolutional layer have stride=1, with n-gram information (1, 2, 3, 4, 5 gram). The learning rate is default from BERT.(Devlin et al., 2019).

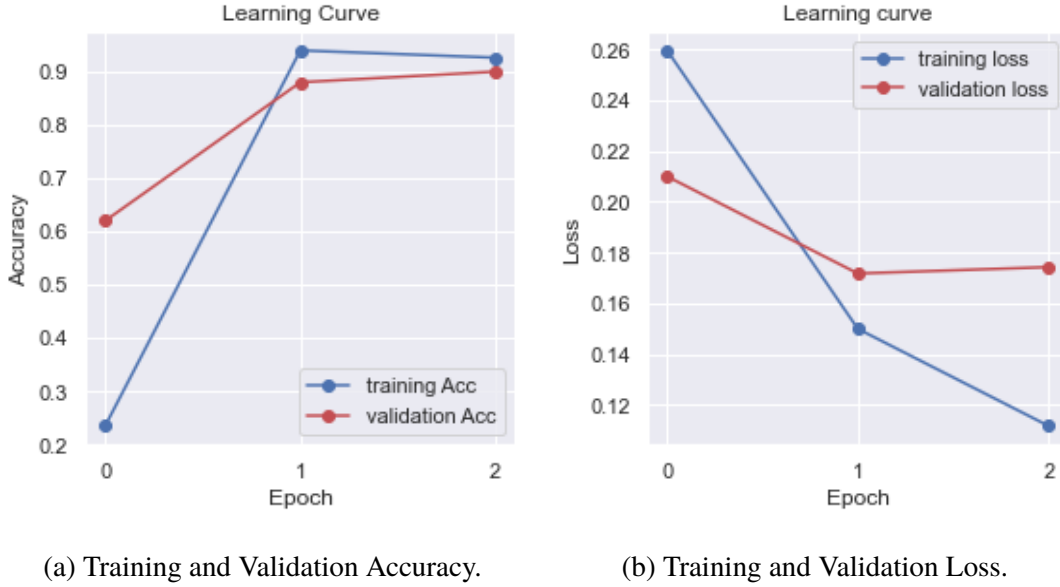


Figure 6.3: Learning Curve for *Dev-multi-lingual-Error-Detection-Model*. (a) Training and Validation Accuracy. (b) Training and Validation Loss.

The above procedure was followed to create five models. Four models were created for individual language namely, *Hindi_Model*, *Konkani_Model*, *Marathi_Model* and *Nepali_Model*, trained with respective languages data. Finally *Dev-multi-lingual-Error-Detection-Model* was created, which is trained with data from all four languages. All five models started overfitting after epoch 1. Figure 6.3 shows the training, validation loss, and accuracy for *Dev-multi-lingual-Error-Detection-Model*.

6.6 Results and Analysis

20% of *Indic-post-OCR-dataset* has been used to evaluate each model. To evaluate the model, True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values are used. TP gives the percentage of errors correctly detected, and TN gives the percentage of correct words that are identified as valid by the model. The FN value shows the percentage of undetected errors. The FP shows the percentage of correct words labeled as errors. An effective error detection system should have good TP without generating much FP. Table 6.5 shows

the TP, TN, FP, and FN for the models. Precision, recall, and F-score for five models are presented in Table 6.6. The language-specific model namely, *Hindi_Model*, *Konkani_Model*,

Model	Testing language	TP	TN	FN	FP
<i>Hindi_Model</i>	Hindi	7.59%	90.19%	0.74%	1.48%
<i>Konkani_Model</i>	Konkani	6.38%	91.03%	0.34%	2.24%
<i>Marathi_Model</i>	Marathi	8.74%	88.55%	1.06%	1.65%
<i>Nepali_Model</i>	Nepali	11.49%	76.53%	6.87%	5.12%
<i>Dev-multi-lingual-Error-Detection-Model</i>	Hindi	7.40%	90.19%	0.93%	1.48%
	Konkani	6.21%	91.03%	0.52%	2.24%
	Marathi	8.74%	88.43%	1.06%	1.77%
	Nepali	11.61%	76.53%	6.99%	5.12%

Table 6.5: TP, TN, FN, and FP of error detection models tested with each of the languages

Model	Testing language	Precision	Recall	F-score
<i>Hindi_Model</i>	Hindi	0.84	0.91	0.87
<i>Konkani_Model</i>	Konkani	0.74	0.95	0.83
<i>Marathi_Model</i>	Marathi	0.84	0.89	0.86
<i>Nepali_Model</i>	Nepali	0.69	0.63	0.66
<i>Dev-multi-lingual-Error-Detection-Model</i>	Hindi	0.83	0.89	0.86
	Konkani	0.73	0.92	0.82
	Marathi	0.83	0.89	0.86
	Nepali	0.69	0.62	0.66

Table 6.6: Precision, Recall and F-score for error detection models

Marathi_Model and *Nepali_Model*, shows an F-score of 0.87, 0.83, 0.86 and 0.66, for Hindi, Konkani, Marathi and Nepali respectively. And the *Dev-multi-lingual-Error-Detection-Model* model, which supports all four languages shows an F-score for all four languages similar to that of the respective language-specific models. These models also show high recall.

Table 6.7 shows examples of the OCR'd word, the actual word, the gloss, the output of the *Dev-multi-lingual-Error-Detection-Model* model, and the actual label for each language. Firstly, it

Language	OCRed word Input	Actual Word	Gloss	Model* output	Actual Label
Hindi	किताबों	किताबों	books	✓	✓
	अन्तम्वस्त्र	अन्तःवस्त्र	undergarments	✗	✗
Konkani	असोच	असोच	just like that	✓	✓
	पयर्लीच	पयर्लीच	already	✗	✗
Marathi	जुन्या	जुन्या	old	✓	✓
	डोळ्यातून	डोळ्यातून	from the eyes	✗	✗
Nepali	थाहा	थाहा	know	✓	✓
	हजारौं	हजारौं	thousands	✗	✗
* Dev-multi-lingual-Error-Detection-Model output					

Table 6.7: Example of *Dev-multi-lingual-Error-Detection-Model* output for each of the languages.

was observed that the model could identify words with apparent errors. Secondly, it correctly identifies the word with error, although it is a valid and frequent word in another language, as shown for in Table 6.8 e.g.1. In this example 1 Konkani sentence was input to the model and one of the incorrectly recognized word was नही [nəhi], which is a valid and frequent word in Hindi. This table has four rows, first with actual text in a specific language such as Konkani, second column is its English translation, third is OCRed text, which is input to the developed model, and the fourth row shows text labeled by *Dev-multi-lingual-Error-Detection-Model*. Here, the green-colored text indicates the word is correct (label 0), and the red-underlined text indicates the word with an error (label 1).

Thirdly, the model correctly classifies words as per context, although trained with four languages, as shown in Table 6.8 e.g. 2 and 3. In Table 6.8 example 2, the word हैं [hē] is recognized as हैं [həē], which is incorrect, and the model correctly classifies it as an error. The same word हैं appears in example 3, where it is correctly recognized by OCR, and the model classifies it as valid. Next example 4 in Table 6.8 the word मोगाळ [mogaḷ] is recognized as मोगा [moga] both are valid words in Konkani, the model correctly identifies the मोगा as incorrect word (refer Table 6.8 e.g.4). Another example in Table 6.8 e.g.5 the Marathi word वेळी [veḷi] was incorrectly recognized word

e.g. 1	Actual text (Konkani) English translation OCRed output <i>Dev-multi-lingual-Error-Detection-Model</i>	रेंजराचीय चूक न्ही. Not the Ranger's fault. रेंजराचीय चूक <u>नही</u>. रेंजराचीय चूक <u>नही</u>.
e.g. 2	Actual text (Konkani) English translation OCRed output <i>Dev-multi-lingual-Error-Detection-Model</i>	हें सगळें आयकुपाची तयारी दवरुंक जाय. You have to be prepared to hear all this. <u>हें सगळें</u> आयकुपाची तयारी दवरुंक जाय. <u>हें सगळें</u> आयकुपाची तयारी दवरुंक जाय.
e.g. 3	Actual text (Konkani) English translation OCRed output <i>Dev-multi-lingual-Error-Detection-Model</i>	पूण हें तो सोंपेपणान मानूंक तयार नासतलो. But he wouldn't be willing to accept this easily. पूण हें तो सोंपेपणान मानूंक तयार नासतलो. पूण हें तो सोंपेपणान मानूंक तयार नासतलो.
e.g. 4	Actual text (Konkani) English translation OCRed output <i>Dev-multi-lingual-Error-Detection-Model</i>	म्हाका मोगाळ आनी जतनाय घेवपी सांगाती जाय. I need loving and caring life partner. म्हाका <u>मोगा</u> आनी जतनाय घेवपी सांगाती जाय. म्हाका <u>मोगा</u> आनी जतनाय घेवपी सांगाती जाय.
e.g. 5	Actual text (Marathi) English translation OCRed output <i>Dev-multi-lingual-Error-Detection-Model</i>	त्या वेळी त्या फळांतील परागकण त्यांच्या शरीराला चिकटतात. That's when the pollen from those fruits sticks to their bodies. त्या <u>वेळी</u> त्या <u>फळांतील</u> परागकण त्यांच्या शरीराला चिकटतात. त्या <u>वेळी</u> त्या <u>फळांतील</u> परागकण त्यांच्या शरीराला चिकटतात.

Table 6.8: *Dev-multi-lingual-Error-Detection-Model* output correctly classify context words (real errors).

वेळी [ved^hi:], which is a valid word in Marathi and model correctly identifies as invalid (refer Table 6.8 e.g.5) Lastly, it was observed that this model correctly classifies punctuation marks in each of the languages as shown in tables 6.8 and 6.9.

A	अहा गजब की बेशरम ठहरी मीम कूलहे मटका के, टंगड़ी दिखा के अंग्रेजी में कह रहीं जनियों से, युअर खसम माई यार करके।
B	अहा गजब की बेशरम ठहरी मीम कूलहे मटका के, टंगड़ी अदिखा के <u>अंग्रेजी</u> में कह रहीं जनियों से, युअर खसम माई यार करके ।
C	<u>अहा</u> गजब की बेशरम ठहरी मीम कूलहे मटका के, <u>टंगड़ी</u> दिखा के अंग्रेजी में कह रहीं जनियों से, युअर खसम माई यार करके ।
Remark अहा (Interjections) अंग्रेजी or अंग्रेजी (Similar Spelling) टंगड़ी (ङ Canonical Equivalence) ट ं ग ड ़ ी (ड (U+0921) + ़ (U+093C) Constituent characters) ट ं ग ड ी (ड (U+095C) Precomposed character)	
A: Actual text (Hindi) B: OCR'd Text C: Dev-multi-lingual-Error-Detection-Model Output	

Table 6.9: Output of *Dev-multi-lingual-Error-Detection-Model* for 3 types of words 1) Interjections 2) Valid words having similar spelling 3) Words having Canonical Equivalence character

Actual text (Code Mixed Konkani Hindi)	सबसे जान प्यारी म्हणटात न्हय?
English translation	Isn't it called sweetheart?
OCR'd output	सबसे जान प्यारी म्हणटात न्हय? .
<i>Dev-multi-lingual-Error-Detection-Model</i> Output	सबसे <u>जान प्यारी</u> म्हणटात न्हय?

Table 6.10: Output of *Dev-multi-lingual-Error-Detection-Model* for sentence having code mixed (Konkani & Hindi).

Let's now look at the model's inadequacies. First, the model could not correctly classify interjections (अहा [əha]) as shown in Table 6.9. Second, it cannot classify words having equivalent canonical characters² ie टंगड़ी [t̪əŋd̪i] as shown in Table 6.9. Third, the incorrectly recognized word अंग्रेजी [əŋɡred̪zi], which has spelling quite similar to actual word अंग्रेजी [əŋɡr̪ed̪zi] is classified as valid by the model as seen in Table 6.9. In this case, the model has labeled the invalid

²Canonical equivalence is a form of equivalence that preserves visually and functionally equivalent characters. <https://www.w3.org/TR/ilreq/>

Actual text (Nepali)	वाह दाजु!
English Translation	Wow brother!
OCRed text	वाह दाजु!.
<i>Dev-multi-lingual-Error-Detection-Model</i>	वाह दाजु!

Table 6.11: Output of error detection *Dev-multi-lingual-Error-Detection-Model* for Nepali.

word as correct in the instance. Fourth, the model does not classify correctly recognized foreign transliterated words as valid unless the word is frequently used within the same domain. The proper noun is classified as invalid if it does not appear in training data. Next, it fails to classify words correctly in a code mixed sentence, as shown in Table 6.10, which contains a sentence in Hindi-Konkani. Example in this sentences सबसे जान प्यारी म्हणटात न्हय? [səbəse dʒan pjari mɦɳəʈats̪ nɦəj]? , the part of sentence is in Hindi i.e. सबसे जान प्यारी [səbəse dʒan pjari] and other part of the sentence is in Konkani म्हणटात न्हय? [mɦɳəʈats̪ nɦəj]?. In this sentence, the OCR-recognized text is correct. However, the model classified some of these words as incorrect. Lastly it does not identify a valid number as an error. For example, the actual text is जमीन २५ and OCRed text is जमीन 5 25. However, the developed model classifies number 5 as a valid text. Otherwise, if the number is invalid, eg. 10०8४, the model correctly classifies it as invalid. In Nepali, one word can be written in a variety of ways. Because of this, error detection in Nepali is more difficult than in other languages (Shahi and Sitaula, 2022). For example, word सक्दिनँ in Nepali can also be written as सक्दैँन, सक्दैँनन् or सक्दिन, which convey the same meaning. As a result, the model identifies the wrongly recognized word सक्दिन [səkdʒin] as valid. Another example in Nepali 'मूल ढोका' [mul dʰoka] and 'मुख्य द्वार' [mukʰj d̪var] are different words with same meaning. Consequently, a word's frequency of occurrence within a corpus may affect how it is classified. Again, the model classifies interjection word वाह [vah] in Nepali as invalid, as shown in Table 6.11.

6.7 Conclusion

This chapter presents a study on the performance evaluation of two Hindi Devanagari OCRs, namely Tesseract and Indsenz. The evaluation is conducted using *Indic-OCR-test-dataset* which supports four languages; Hindi, Konkani, Marathi and Nepali. The findings indicate that Tesser-

act OCR outperforms Indsenz OCR across all four languages. Consequently, Tesseract OCR is selected as the basis for creating the training and test dataset called *Indic-post-OCR-dataset*. *Dev-multi-lingual-Error-Detection-Model* was created using the *Indic-post-OCR-dataset*, employing mBERT with CNN techniques. The model's performance is evaluated on the four Indian languages mentioned above. Additionally, separate models are created for each language. The test results reveal that the F-score of the language-specific models is similar to that of the *Dev-multi-lingual-Error-Detection-Model*. The study demonstrates the potential of leveraging existing tools designed for resource-rich languages to provide solutions for low-resource languages. A model was successfully developed that supports all four languages. Error analysis indicates that the model accurately identifies real errors but struggles with classifying interjections and code-mixed sentences. Furthermore, the model faces challenges in detecting words from different domains, particularly English words that have been transliterated. Addressing these shortcomings requires further research.

This page is intentionally left blank.

Chapter 7

Multi-Font Text Recognizer for Hindi

7.1 Background and Motivation

OCR supporting multi-font text is essential as the data is available in different fonts. The performance study of OCRs showed a difference in the OCR's accuracy for each font (refer Chapter 5; Section 5.3). However, there is no study showing the distribution of these errors with respect to their types. Additionally, there is no multi-font OCR available for Hindi. With the error correction approach, it was possible to handle insertion errors but not very successfully handle deletion errors (discussed in Chapter 5; Section 5.7). Thus, the objective is to reduce insertion and deletion errors for multi-font text images.

The chapter is organized as follows; Section 7.2 which creates training and testing dataset for five fonts. Section 7.3 details the performance of existing OCRs on text in five fonts. Section 7.4 builds multi-font text recognizer for Hindi. Section 7.5 discusses the results. Section 7.6 concludes the chapter.

7.2 Data creation for text line in five fonts

Different properties are considered while choosing the fonts, such as vertical terminals with serifs, horizontal and left swooshes, and variations in grey value with light and dark tones. The *Nakula*, *Baloo*, *Dekko*, *Biryani*, and *Aparajita* fonts (see Figure 7.1 and refer to Section 1.2 for the importance of fonts) with regular styles are selected based on these font properties. These five different fonts are used to create training data using text from the corpus available at Centre for Indian Language Technology ([cfilt.iitb](http://cfilt.iitb.ac.in), 2023). The Python script was used to create images

वह कथा या कविता बदल जाती है
वह कथा या कविता बदल जाती है
वह कथा या कविता बदल जाती है
वह कथा या कविता बदल जाती है
वह कथा या कविता बदल जाती है

Figure 7.1: Sample text in fonts (from top to bottom): *Nakula*, *Baloo*, *Dekko*, *Biryani* and *Aparajita*

of each line in the text corpus for fonts. The training data is created for all five fonts, namely *Nakula*, *Baloo*, *Dekko*, *Biryani*, and *Aparajita*. It consists of an image file of each line in text document images and its corresponding text. Training data consists of 500K images and the same number of text files. This training data is named as *Multi_Font_Train* dataset. To create test data, the text from standard Dataset IIITH ([Mathew et al., 2016](#)) is used, which consists of 100 image files and corresponding text files. This data creation involved generating text lines in five different fonts. The test data is created using the following steps:

1. Created text files named as *Text_Files_TestDataset* for each text line in these 100 text files of standard dataset. There are 2812 text files in *Text_Files_TestDataset*.
2. For each of the file in *Text_Files_TestDataset* it creates corresponding text document images for *Nakula* font using python script, which take input as text and font style and output is same text as document image in ".png" format. This is named as BMT_*Nakula* dataset, which has 2812 files in ".png" format.
3. Like step 2, created the test dataset for *Baloo* font named as BMT_*Baloo* dataset consisting of 2812 files in ".png" format.
4. Similarly, created test dataset for *Dekko* font named as BMT_*Dekko*, *Biryani* font named as BMT_*Biryani* and *Aparajita* font named as BMT_*Aparajita*.

7.3 Performance of existing OCRs

The performance of the three most popular Devanagari OCR systems, namely Tesseract, Indsenz, and e-Aksharayan OCR, on text document images in different font, specifically *Nakula* and *Aparajita*, is discussed here. The performance results of these OCR systems are shown in Table 7.1. These results demonstrate that OCR accuracy is dependent on the fonts used. OCR error

Font/OCR	Tesseract	Indsenz	e-Aksharayan
<i>Nakula</i>	90.14%	81.74%	70.14%
<i>Aparajita</i>	77.27%	33.21%	22.20%

Table 7.1: Performance of OCR's Character Accuracy for text in different font.

Type of ErrorOCR	Tesseract	Indsenz	e-Aksharayan
Insertion	9.01%	5.17%	15.07%
Substitution	9.54%	41.52%	55.32%
Deletion	3.37%	19.34%	7.39%

Table 7.2: Error analysis of OCR's for text in *Aparajita* font.

Type of Error/OCR	Tesseract	Indsenz	e-Aksharayan
Insertion	4.27%	5.59%	10.58%
Substitution	3.53%	7.49%	33.19%
Deletion	1.86%	4.20%	8.09%

Table 7.3: Error analysis of OCR's for text in *Nakula* font.

types were examined to understand the reason for different accuracy. OCR typically produces three types of errors: insertion, deletion, and substitution. These errors are analyzed using the OCR evaluation tool ([Rice and Nartker, 2022](#)). Table 7.2 shows the percentage of type errors for document images in *Aparajita* font, and Table 7.3 shows the percentage of type errors for document images in *Nakula* font. It was observed from this that insertion and deletion errors

differ for these two fonts. This shows that the type of OCR error differs for each font. Handling the insertion and deletion type of error in the post-processing OCR output is difficult. This work attempts to minimize insertion and deletion type errors and to create OCR, which shows consistent accuracy across all font.

7.4 Experimental setup for Multi-Font Text Recognizer using LSTM neural networks

OCRopus ([OCRopy, 2023](#)) is an open-source document analysis and recognition tool utilized to create the Multi-Font OCR. It uses Recurrent Neural Networks (RNN) for recognition, specifically a 1D LSTM architecture. In this architecture, a window with a height equal to that of the input image (referred to as the "depth of the sequence") and a width of 1 pixel (called a "frame") moves over the text line. These 1D sequences (frames) are then fed to the input layer of the LSTM network.

To achieve normalization, it is important to ensure that the sequence depth is consistent across all input text-line images. The system is trained with the *Multi_Font_Train* dataset. The LSTM Recurrent Neural Network learns the sequence mapping of Hindi text in different fonts. The model has 48 inputs, 200 nodes in a hidden layer, and 249 outputs. The inputs to the network are columns of pixels, which are fed into the network one at a time from left to right. The outputs are scores for each possible letter. The learning rate and momentum were set to standard values of 0.0001 and 0.9, respectively. The final model was created after 800,000 iterations. This was named as MFD_OCR.

7.5 Results and Analysis

7.5.1 Accuracy of OCR's on text document images of different font

ISRI analytic tool is used to evaluate OCRs ([Rice and Nartker, 2022](#)). OCR's performance was evaluated with the created test dataset for all five fonts namely BMT_Nakula, BMT_Baloo, BMT_Dekko, BMT_Biryani and BMT_Aparajita. The performance of OCR's for test dataset is

shown in Table 7.4. The best character accuracy for BMT_Nakula, BMT_Baloo and BMT_Aparajita

Dataset/OCR	Tesseract	Indsenz	e-Aksharayan	MFD_OCR
BMT_Nakula	90.14%	81.74%	70.14%	89.23%
BMT_Baloo	89.64%	78.15%	62.93%	88.49%
BMT_Dekko	77.87%	72.34%	47.51%	89.03%
BMT_Biryani	88.56%	77.28%	59.41%	88.89%
BMT_Aparajita	77.27%	33.21%	22.20%	69.87%

Table 7.4: OCR's Character Accuracy on test dataset

test dataset is shown by Tesseract OCR. The proposed MFD_OCR performs better for BMT_Dekko and BMT_Biryani test dataset compared to all other OCR's. The Indsenz and eAksharayan OCR's showed very low accuracy for BMT_Aparajita test dataset. It has been observed that each OCR accuracy varies for each font, but MFD_OCR obtained consistent accuracy across all the fonts.

7.5.2 Error analysis for Tesseract OCR, Indsenz OCR, e-Aksharayan OCR and MFD_OCR for each test dataset.

Performed the error analysis of all OCR's with the help of OCR evaluation tool ([Rice and Nartker, 2022](#)). The error in OCR is due to insertion, substitution and deletion of the character. These errors are discussed in detail for each of the test dataset.

Type of Error/OCR	Tesseract	Indsenz	e-Aksharayan	MFD_OCR
Insertion	4.27%	5.59%	10.58%	4.91%
Substitution	3.53%	7.49%	33.19%	5.55%
Deletion	1.86%	4.20%	8.09%	0.86%

Table 7.5: Percentage of OCR's errors for BMT_Nakula test dataset.

Table 7.5 shows the percentage of errors for each of the OCR for BMT_Nakula test dataset. The insertion and substitution type of error are less in Tesseract OCR for BMT_Nakula dataset. The

MFD_OCR has less deletion type of errors and insertion type of error is nearly the same as that of Tesseract OCR.

Type of Error/OCR	Tesseract	Indsenz	e-Aksharayan	MFD_OCR
Insertion	4.98%	7.78%	20.19%	4.58%
Substitution	2.72%	8.46%	14.24%	5.39%
Deletion	2.10%	3.90%	2.62%	0.90%

Table 7.6: Percentage of OCR's errors for BMT_Baloo test dataset.

The Table 7.6 shows the percentage of errors for each of the OCR for BMT_Baloo dataset. The insertion and deletion type of error are less in MFD_OCR for the BMT_Baloo dataset. The Tesseract OCR has less substitution type of errors.

Type of Error/OCR	Tesseract	Indsenz	e-Aksharayan	MFD_OCR
Insertion	6.05%	7.07%	6.50%	2.23%
Substitution	7.71%	12.60%	30.35%	6.79%
Deletion	7.86%	7.19%	15.62%	1.49%

Table 7.7: Percentage of OCR's errors for BMT_Dekko test dataset.

Table 7.7 shows the percentage of errors for each of the OCR for BMT_Dekko dataset. The insertion, substitution and deletion type of error are less in MFD_OCR for BMT_Dekko dataset.

Type of Error/OCR	Tesseract	Indsenz	e-Aksharayan	MFD_OCR
Insertion	4.29%	5.63%	4.56%	4.42%
Substitution	4.09%	10.75%	23.37%	4.94%
Deletion	2.54%	5.22%	12.65%	1.00%

Table 7.8: Percentage of OCR's errors for BMT_Biryani test dataset.

Table 7.8 shows the percentage of errors for each of the OCR for BMT_Biryani test dataset. The Tesseract OCR has fewer substitution and insertion type errors. The MFD_OCR has less deletion

type of errors and insertion, and substitution type of error is nearly the same as that of Tesseract OCR. Table 7.9 shows the percentage of errors for each of the OCR for BMT_Aparajita test

Type of Error/OCR	Tesseract	Indsenz	e-Aksharayan	MFD_OCR
Insertion	9.01%	5.17%	15.07%	6.24%
Substitution	9.54%	41.52%	55.32%	22.61%
Deletion	3.37%	19.34%	7.39%	1.40%

Table 7.9: Percentage of OCR's errors for BMT_Aparajita test dataset.

dataset. The insertion type of error are less in Indsenz OCR for BMT_Aparajita dataset. The MFD_OCR has less deletion type of errors, and The Tesseract OCR has less substitution type of errors.

7.5.3 Percentage of types of errors for each of the OCR's for different types of fonts style

Figure 7.2 plots the percentage of deletion type of errors for each test dataset versus OCRs. It is observed from Figure 7.2 that the proposed MFD_OCR gives less deletion errors for all types of font, which has been tested with a particular type of dataset. Figure 7.3 plots the percentage of insertion type errors for each test dataset versus OCRs. Figure 7.3 showed Multi-Font OCR has consistent insertion type of error for all fonts, which has been tested with a particular type of dataset. Figure 7.4 plots the percentage of substitution type of errors for each test dataset versus OCRs. Finally, substitution types of error are also consistent for all fonts in MFD_OCR, except for text in Aparajita font.

7.6 Conclusion

This work studies the performance of existing Hindi OCR on two fonts *Nakula* and *Aparajita* and analysed the type of errors. It was observed that types of OCR errors are not consistent for these two fonts. This work has contributed to the design of multi-font OCR for Hindi (MFD_OCR), supporting five fonts using LSTM (open-source OCRopus). Results showed that, MFD_OCR

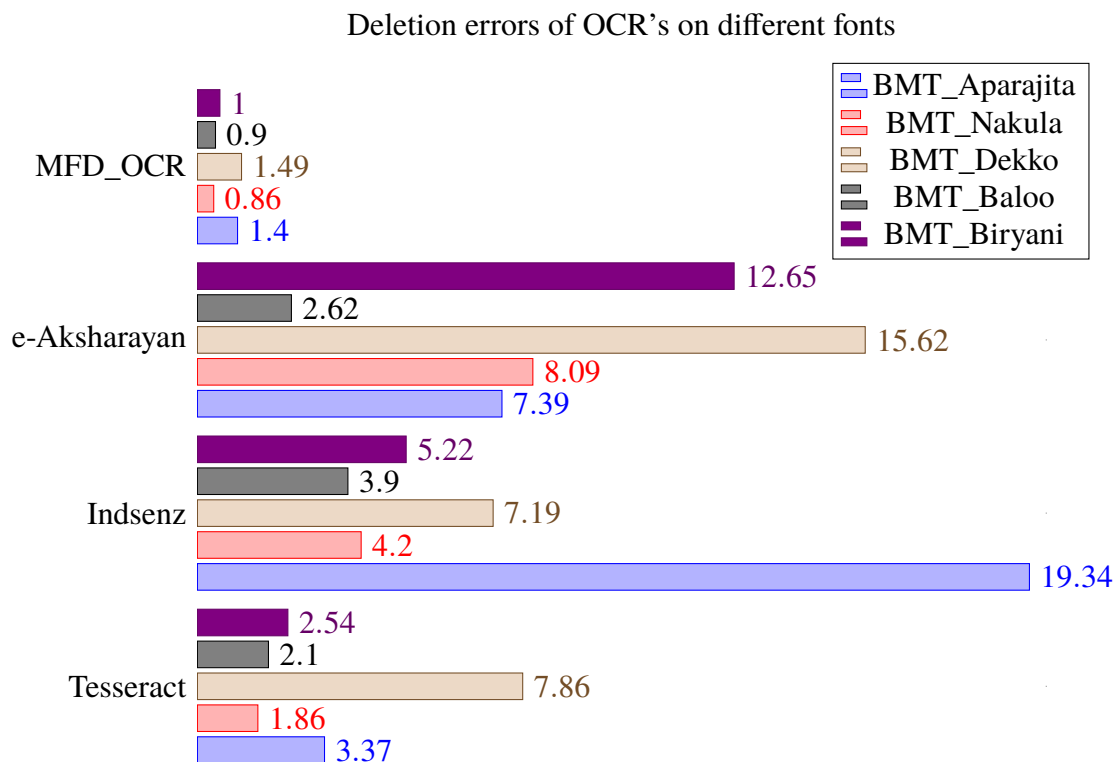


Figure 7.2: Deletion type of errors of OCR's for text document in different fonts

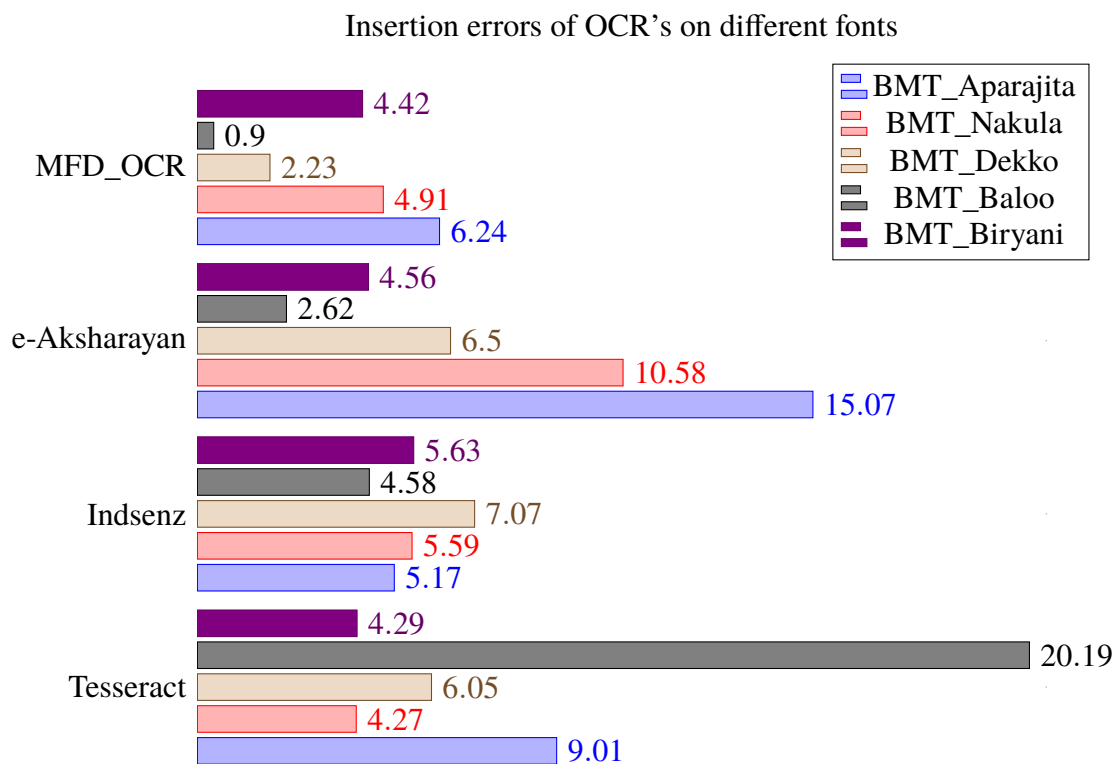


Figure 7.3: Insertion type of errors of OCR's for text document in different fonts

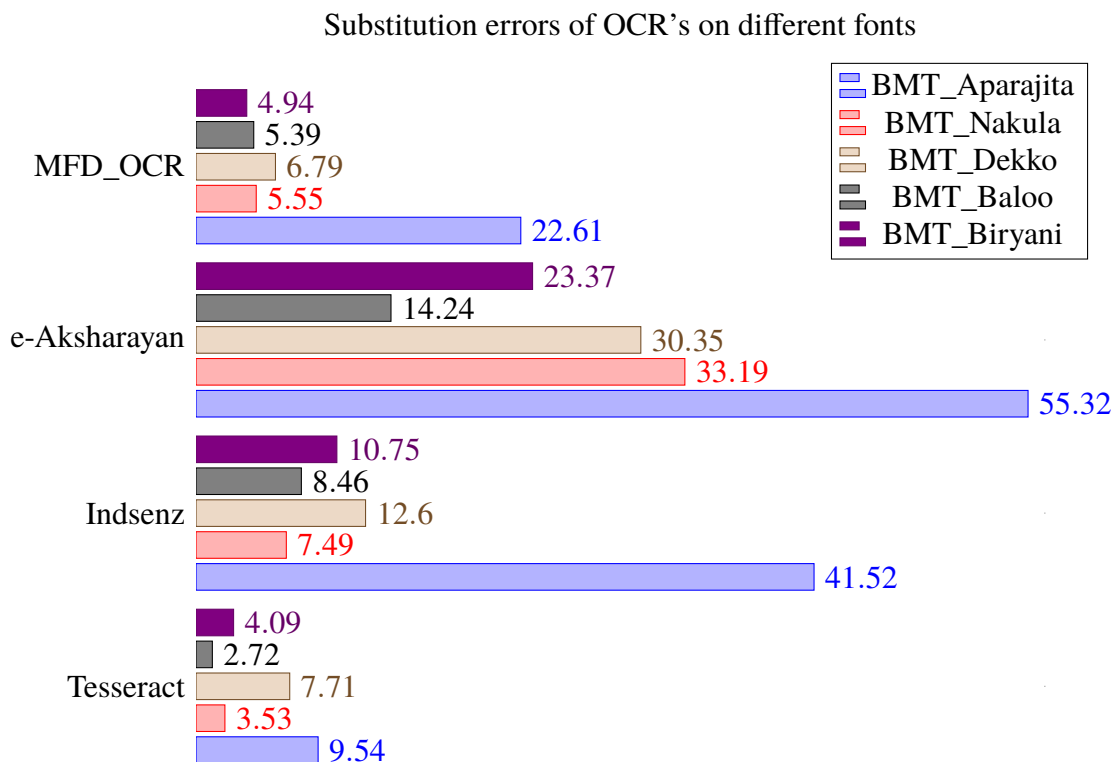


Figure 7.4: Substitution type of errors of OCR's for text document in different fonts

demonstrated consistent accuracy across all five test datasets and achieved comparatively good accuracy for text in the *Dekko* and *Biryani* fonts. Unlike other existing Hindi OCRs, the insertion and deletion errors for the MFD_OCR were consistent for all five fonts; *Nakula*, *Baloo*, *Dekko*, *Biryani* and *Aparajita*. It showed negligible deletion errors i.e. less than 1.4%. Additionally, Tesseract OCR had the lowest substitution error rate compared to all other OCRs and achieved the highest accuracy for text in the *Nakula*, *Baloo*, and *Aparajita* fonts.

This page is intentionally left blank.

Chapter 8

Multi-Font OCR for Konkani

8.1 Background and Motivation

Konkani is primarily spoken along the west coast of India, specifically in the state of Goa, where it is an official language. Given its relatively lower resource availability compared to more widely spoken Indian languages such as Hindi, developing an OCR for Konkani can be a valuable endeavor. To create digital corpus for Konkani language, the digitization of Konkani books and newspaper articles can be of great help. However, the written text is often found to occur in multiple fonts, as shown in Figure 8.1, a sample clipping from Konkani local newspaper *Bhaangarbhuin*¹. This clipping has text in different fonts. Because of this, having OCR compatibility for several fonts is crucial.

The accuracy of NLP tools depends on the availability of a vast and good-quality corpus. Developing NLP tools for Konkani can be speeded up by creating a huge corpus for Konkani. An OCR supporting multiple fonts can help us to achieve this objective. There has been limited exploration in the field of OCR with multiple fonts.

This research has highlighted that usage of characters, *ligatures*, and syllables (akshara) differs for each language (refer Chapter 3, Section 3.8) and its appearance differs for different fonts. OCR accuracy differs for different font (refer Chapter 5; Section 5.3). There is need to develop multi-font Konkani OCR (refer chapter 5; Section 5.6.5). As an open-source solution, Tesseract OCR has demonstrated acceptable accuracy for Hindi (refer Chapter 5; Section 5.3 and Chapter 7, Section 7.5), compared to other available OCRs. Tesseract version 4.0 supports data models for both Latin and Indian languages. It also supports model for the Devanagari script. When it

¹<https://epaper.thegoan.net/3755632/BHAANGARBHUIN/BHAANGARBHUIN#page/1/1>

आंगले उच्च माध्यमिकांत गुणेस्त विद्यार्थ्यांचो भोवमान

■ भांगरभूंय | प्रतिनिधी

Font 1

क्लासिच्यो सुविधां उपलब्ध नासून लेगीत
मेळिल्ल्या येसा विशीं अध्यक्ष प्रशांत नायक

तशेंच राज्यशास्त्र विशयांत 100 गुण घेवन
बोर्डांत फ्यलें आयिल्लें पलक रिबणकार,

उसगांवां 'हीट अँड रन'

Font 2

■ भांगरभूंय | प्रतिनिधी

फोंडें: पार-उसगांवां एमआरएफ कंपनी लागसार भिरांकूळ
अपघातांत आनंद धर्मा नायक (57, पार-उसगांव) ह्या पादचाऱ्याक
मरण आयलें. अपघाता उपरांत ट्रका पोंदा चिड्डिल्या नायक
हांची तकली एक किमी. अंतराचेर आशिल्ल्या एमआरएफ

जीपीचो स्फुटरीक
धपको दोग जखमी

दुसऱ्या एका अपघातांत काल
उसगांवां एमआरएफ कंपनी
लागसारव्या म्हामार्गा वयल्यान वचपी

पर्वी जिपी पोंदा चिड्डिल्यान
ट्रक चालकाक मरण

पातळ्ळां. बुधवारा एकाच दिसांत

Font 3

तीन खुन्यांच्या घडणुकांनी गोंयांत खवदळ पडला. बुधवारा
सकाळीं दिवचले सेतू संगमाचेर बेळगांवच्या तरणाट्याची
कूड मेळ्ळी. सुरो तोंपून ताचो खून केलो. बाणस्तारी मार्केटा
लागसार एके बायलेची कूड मेळ्ळी. गळो अंदून तिचो खून
केलो. माडेल- फातोड्ड्यां एका कामगाराचो तकलेर फातर

Figure 8.1: Bhaangarbhuiin Konkani newspaper articles

comes to creating models using Tesseract, there are generally two primary methods:

- (a) Training from Scratch: This method involves training a model entirely from the ground up. It typically requires a substantial amount of training data, including images of text and corresponding transcriptions. Training from scratch is a resource-intensive process often used when there are no existing models or when the target script or language differs substantially from existing models.
- (b) Training from existing models: This involves taking an existing model with similar script or language characteristics and adapting it for a specific use case. This method is less data-intensive compared to training from scratch. It's a more desirable approach for languages or scripts that are closely related to those for which models already exist.

For less-resourced languages, fine-tuning an existing model that supports a similar script can be a practical and effective way to create OCR solutions. Second approach is used along with

the *Devanagari.unicharset*⁵ and *Devanagari.traindeddata*² model to provide an OCR model for Konkani.

This chapter is organized as follows: Section 8.2 provides details of training and test datasets. Section 8.3 shows Devanagari Tesseract OCR's performance on the Konkani test dataset with ten fonts. Section 8.4 discusses the structure of proposed solution to create multi-font OCR for Konkani. Section 8.5 describes the procedure for creating multi-font OCR for the Konkani language. Section 8.6 discusses the results. Section 8.7 provides the details of Konkani OCR tool interface. Section 8.8 concludes the chapter.

8.2 Datasets

A gold standard Konkani raw text corpus was taken from LDC-IL (Ramamoorthy et al., 2019) containing 3619017 words. The dataset has been divided into two subsets: training and test data. The test data, comprising 27,067 words, has been selected in inline with the available Hindi OCR dataset from IIITH (Mathew et al., 2016). The remaining data has been used as training data consisting of 3591950 words. The procedure used to create the Konkani training dataset is as follows:

- (i) The training data has been merged in one file named *merged_text*. This file has one word on each line.
- (ii) Furthermore, this *merged_text* file is input, Nakula font and 12 pt font size is input to the procedure *text2img-doc* (refer Chapter 4; Section 4.3.1), thus creating *N_text* and *N_Img* files.
- (iii) Step (ii) is repeated for nine other fonts, namely *Amiko*, *Lohit Devanagari*, *Biryani*, *Cam-bay*, *Eczar*, *Gargi*, *Asar*, *Baloo*, and *Aparajita*

This dataset is named *Konkani_train_dataset*, which consists of *N_text* and *N_Img* files in the Konkani language for each font. Thus, total image and text files are 10 times of *N_text* and *N_Img* files. The *Konkani_test_dataset* (created in Section 4.3.5) is used as test dataset.

²<https://github.com/tesseract-ocr/tessdata/tree/main>

8.3 Performance of Devanagari Tesseract OCR on Konkani test data having 10 fonts *Konkani_test_dataset*

Sr. No.	Font name	Sample text
1	<i>Amiko</i>	हांवें एका पुस्तकांत वाचिल्लें
2	<i>Lohit Devanagari</i>	हांवें एका पुस्तकांत वाचिल्लें
3	<i>Biryani</i>	हांवें एका पुस्तकांत वाचिल्लें
4	<i>Nakula</i>	हांवें एका पुस्तकांत वाचिल्लें
5	<i>Cambay</i>	हांवें एका पुस्तकांत वाचिल्लें
6	<i>Eczar</i>	हांवें एका पुस्तकांत वाचिल्लें
7	<i>Gargi</i>	हांवें एका पुस्तकांत वाचिल्लें
8	<i>Asar</i>	हांवें एका पुस्तकांत वाचिल्लें
9	<i>Baloo</i>	हांवें एका पुस्तकांत वाचिल्लें
10	<i>Aparajita</i>	हांवें एका पुस्तकांत वाचिल्लें

Table 8.1: Sample Konkani text in ten fonts

This section shows the performance of the existing Devanagari Tesseract OCR on the dataset with ten fonts. In this study, ten fonts are selected, namely *Amiko*, *Lohit Devanagari*, *Biryani*, *Nakula*, *Cambay*, *Eczar*, *Gargi*, *Asar*, *Baloo* and *Aparajita*. These fonts are selected based on their usage in Konkani books, newspapers, and online articles. The *Nakula*, *Amiko*, *Lohit Devanagari* fonts are commonly employed for the Devanagari script. *Nakula* is known for its curved characters and slight variations in stroke thickness³. *Nakula* font aligns with the writing style observed in the existing Hindi OCR evaluation dataset (Mathew et al., 2016; Kundaikar and Pawar, 2020) and in Konkani books. Also, *Amiko* font is observed in Konkani books. In addition to these, other fonts, such as *Cambay*, *Eczar*, *Gargi*, *Asar*, *Baloo* and *Aparajita* have been chosen to support various types of Konkani content, such as newspaper articles and poetry books. Furthermore, fonts namely *Lohit Devanagari*, *Nakula*, and *Gargi* are supported in currently used applications like L^AT_EX for research article writing. Table 8.1 shows Konkani text in ten fonts. It is important to note here that the character appearance and the representation of conjunct consonants differs based on font. E.g., the conjunct consonant झ in word वाचिल्लें

³<https://bombay.indology.info/software/fonts/devanagari/index.html>

appears differently for font *Amiko* and *Cambay*. Devanagari Tesseract OCR was tested using

<i>Konkani_test_dataset</i> with font	Devanagari Tesseract OCR	
	Word Accuracy	Character Accuracy
<i>Amiko</i>	95.71%	97.39%
<i>Lohit Devanagari</i>	95.45%	97.30%
<i>Biryani</i>	95.21%	97.09%
<i>Nakula</i>	94.89%	96.85%
<i>Cambay</i>	94.74%	96.20%
<i>Eczar</i>	91.50%	96.09%
<i>Gargi</i>	93.66%	96.85%
<i>Asar</i>	89.55%	94.07%
<i>Baloo</i>	90.12%	95.07%
<i>Aparajita</i>	83.67%	93.61%
Average	92.43%	96.05%

Table 8.2: Performance of Devanagari Tesseract OCR on multiple fonts

Konkani_test_dataset. The performance of Devanagari Tesseract OCR on *Konkani_test_dataset* for each of the ten fonts is shown in Table 8.2. It is observed that accuracy differs for groups of fonts. For example, a dataset with font *Amiko*, *Lohit Devanagari*, *Biryani* has a word accuracy of 95%. The dataset with font *Nakula* and *Cambay* has 94% word accuracy, and *Gargi* shows 93% word accuracy. *Baloo* and *Asar* showed 90% word accuracy. *Konkani_test_dataset* with font *Aparajita* showed 83% word accuracy. For Konkani, the primary sources of printed data are newspaper articles and books. However, the text in these newspapers often uses multiple fonts. Consequently, it is essential to have an OCR that can handle multiple fonts effectively. This capability is crucial for future developments, similar to OCR for English, which supports recognizing text in multiple fonts.

Table 8.3 lists top 10 errors exhibited by Tesseract Devanagari model. It is observed that top most errors are because of consonant र. And third most errors are because of consonant ल. The objective is to reduce these consonant errors and build OCR for Konkani supporting multi-font.

Sr. No.	Devanagari Tesseract OCR	
	Actual character	Error
1	र Consonant [rə]	0.112%
2	' (U+2019)(Right Single Quotation mark)	0.101%
3	ल Consonant	0.057%
4	' (U+2018) (Left Single Quotation mark)	0.055%
5	“ Left double quotation mark	0.053%
6	? Question mark	0.039%
7	ँ (U+0945) Candra E	0.034%
8	व Consonant	0.024%
9	ँ Independent vowel	0.017%
10	ँ Devanagari vowel sign	0.008%

Table 8.3: Top 10 substitution errors exhibited by Devanagari Tesseract OCR on *Konkani_test_dataset* for *Amiko*, *Biryani*, *Nakula*, *Cambay*, *Eczar*, *Gargi* and *Asar* fonts

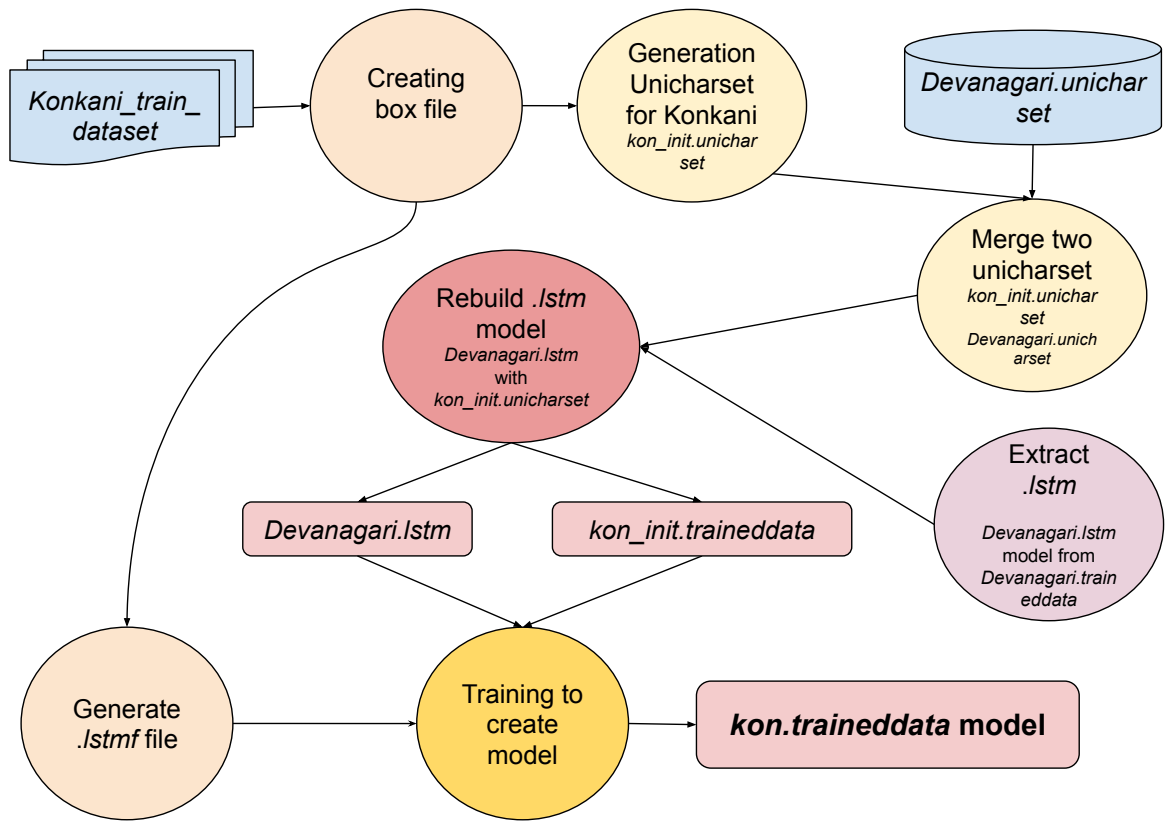
8.4 Structure of proposed solution

This work provides an OCR solution for less-resourced Konkani using Tesseract LSTM composed in the Devanagari. This approach extends support for recognizing characters in multiple fonts.

8.5 Experimental set up to create OCR for Konkani using Tesseract LSTM

The procedure for creating an multi-font OCR model for Konkani language involves the following steps, and the block diagram of same is shown in Figure 8.2.

- (i) Dataset Used: The *Konkani_train_dataset* as described in section 8.2 is utilized here for training purpose. Further, box files are created for each of the images.
- (ii) Generation of character set for Konkani (i.e. *.unicharset* file): Unicharset files are gener-

Figure 8.2: Procedure to create *kon.traineddata* model

ated based on the box files specific to *Amiko* font. Obtaining the character set from the data for any single font is adequate. Moreover, the *Amiko* font exhibited a distinct appearance for Devanagari conjunct consonants. This step involves creating a character set representing the characters in the training data. This unicharset is named as *kon_init.unicharset*.

- (iii) Extract lstm model from *Devanagari.traineddata*² model: The .lstm model is extracted from *Devanagari.traineddata* using *combine_tessdata*⁴. Thus it generates *Devanagari.lstm* model.
- (iv) Merge character set file (i.e. .unicharset file): The character sets for *Devanagari.unicharset*⁵ and *kon_init.unicharset* are merged using *merge_unicharsets*⁶. This step combines the characters from these languages to create a unified character set that can accommodate characters from both .unicharset files. This file is named as *kon.unicharset*.

⁴https://github.com/tesseract-ocr/tesseract/blob/main/doc/combine_tessdata.1.asc

⁵<https://github.com/tesseract-ocr/langdata>

⁶https://github.com/tesseract-ocr/tesseract/blob/main/doc/merge_unicharsets.1.asc

- (v) Rebuild *Devanagari.lstm* model: The *Combine_lang_model*⁷ takes *kon.unicharset* file generated in step (iv). Further, it rebuilds the *Devanagari.lstm* model (created in step (iii)). It also generates a starter traineddata file, e.g., *kon_init.traineddata* that serves as a base for training a Long Short-Term Memory (LSTM)-based neural network model for OCR.
- (vi) Generate training files: The *.lstmf* files are generated for each of the box file. This is done by setting specific parameters such as page segmentation mode *psm*⁸ is set to 6, and OCR engine mode (*oem*) is set to 1, for the *lstm.train* command. The path to these files are appended to a text file named as *training_files.txt*
- (vii) Training with pretrained Model⁹: Training is initiated using *lstmtraining* command with the parameter *Devanagari.lstm* model generated in step (v) as a base. Along with the *training_files.txt* generated in step (vi) and *kon_init.traineddata* generated in step (v) as a starter model. The objective is to adapt this model to the characteristics of Konkani. This step results in the creation of the *kon.traineddata* model specifically tailored for Konkani. Training parameter such as *psm* is set to 6. And (*oem*) is set to 1. Debug-Interval is set to 0. The max-iteration is set to 10,000. Further, it generates checkpoints periodically written as the model with the minimum training error at that point in training.
- (viii) The best checkpoint is converted to OCR model named as *kon.traineddata*, which is also referred to as *Konkani_Multi_Font* OCR.

8.6 Results and Discussion

This section has four parts: Section 8.6.1 discusses the performance of *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for each font. Section 8.6.2 delves into the detailed errors generated by *Konkani_Multi_Font* OCR on *Konkani_test_dataset*. Section 8.6.3 compares the top 10 substitution errors exhibited by Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR. Section 8.6.4 shows the performance of *Konkani_Multi_Font* OCR on Konkani books and newspaper articles (real data).

⁷https://github.com/tesseract-ocr/tesseract/blob/main/doc/combine_lang_model.1.asc

⁸<https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html>

⁹<https://github.com/tesseract-ocr/tesseract/blob/main/doc/lstmtraining.1.asc>

The evaluation metrics used to measure the performance of the OCR are Word accuracy (WA), Character accuracy (CA), Character Error Rate (CER). OCR evaluation tool ([Rice and Nartker, 2022](#)) has been used to calculate OCR's word and character accuracy. Additionally, the *wor-daccsum* and *accsum* commands are used to calculate average word and character accuracy and obtain details of insertion, substitution, and deletion errors ([Rice and Nartker, 2022](#)).

8.6.1 Performance of *Konkani_Multi_Font* OCR on *Konkani_test_data*

<i>Konkani_test_dataset</i> with font	<i>Konkani_Multi_Font</i> OCR	
	Word accuracy	Character accuracy
<i>Amiko</i>	97.22%	97.91%
<i>Lohit Devanagari</i>	97.22%	97.91%
<i>Biryani</i>	97.22%	97.91%
<i>Nakula</i>	97.09%	97.93%
<i>Cambay</i>	97.02%	97.65%
<i>Eczar</i>	95.68%	97.08%
<i>Gargi</i>	95.63%	97.48%
<i>Asar</i>	95.00%	96.81%
<i>Baloo</i>	94.99%	96.42%
<i>Aparajita</i>	92.34%	96.17%
Average	95.94%	97.33%

Table 8.4: Performance of *Konkani_Multi_Font* OCR on *Konkani_test_dataset*

For ten fonts, the *Konkani_Multi_Font* OCR has been tested with *Konkani_test_dataset*. Table 8.4 displays the word and character accuracy of *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for each of the font. It is observed that *Konkani_Multi_Font* OCR showed word accuracy of 97% on test dataset with *Amiko*, *Lohit Devanagari*, *Biryani*, *Nakula* and *Cambay* font. Further, it showed word accuracy of 95% on *Eczar*, *Gargi* and *Asar* font. Additionally, the OCR showed word accuracy of 94.99% and 92.34% on *Baloo* and *Aparajita* font respectively. On average, *Konkani_Multi_Font* OCR showed word accuracy as 95.94% and character accuracy as 97.33% on *Konkani_test_dataset* (refer Table 8.4).

Table 8.2 shows the word and character accuracy of existing Devanagari Tesseract OCR on *Konkani_test_dataset* for ten fonts. The average word and character accuracy for Devanagari Tesseract OCR is 92.43% and 96.05% respectively. It is vital to note *Konkani_Multi_Font* OCR showed consistent word and character accuracy for a group of fonts. On average, there is an improvement in word accuracy by 3.51% compared to the Devanagari Tesseract OCR word. Table 8.5 illustrates the output of both Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR on *Konkani_test_dataset* using *Amiko* font, which yielded the highest word accuracy for both OCRs. It has been observed that *Konkani_Multi_Font* OCR accurately recognizes the words that Devanagari Tesseract OCR could not.

Actual word	Devanagari Tesseract OCR output	<i>Konkani_Multi_Font</i> OCR output
आशिल्लों	आशिष्लों	आशिल्लों
वच्चें	वच्च	वच्चें
विचाल्लें	विचाह्लें	विचाल्लें
सँझ	सँझ	सँझ
भर्जी	भर्जी	भर्जी
वेगवेगळीं	वेगवेगळी	वेगवेगळीं
भ्रश्टाचाराच्या	भ्रश्टाचाराच्या	भ्रश्टाचाराच्या
रँड	रं	रँड
कंपनींत	कंपनीत	कंपनींत
तीय	तीय	तीय
थारायल्लें	थारायहल्लें	थारायल्लें
थाल्लें	याहल्लें	थाल्लें

Table 8.5: Output of Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *Amiko* font

8.6.2 Error analysis of *Konkani_Multi_Font* OCR

A detailed error analysis of *Konkani_Multi_Font* OCR is presented. This has two sub-parts: Section 8.6.2.1 provides details regarding the types of errors exhibited by the OCR (i.e., insertion substitution and deletion) and its percentage. Section 8.6.2.2 provides details of different

substitution errors.

8.6.2.1 Type of OCR errors: insertion substitution and deletion

Usually, the OCR character errors can be categorized into three main types based on the nature of the error: (a) Insertion errors, (b) Substitution errors, and (c) Deletion errors. Table 8.6 shows the percentage of insertion, substitution, deletion errors and CER shown by *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for each of the font. It is observed that *Konkani_test_dataset* with font, *Baloo* and *Aparajita*, are exhibiting higher substitution errors compared to the other eight fonts. Hence, *Konkani_test_dataset* was divided into two categories on the basis of substitution errors. The first category includes test dataset with font *Amiko*, *Lohit Devanagari*, *Biryani*, *Nakula*, *Cambay*, *Eczar*, *Gargi*, and *Asar*, which is named as *Category-I*. The second category consists explicitly of *Konkani_test_dataset* that employ the *Baloo* and *Aparajita* font, which is named as *Category-II*. The average CER exhibited by *Konkani_Multi_Font* OCR is 2.67% (refer Table 8.6 last row). The average percentage for each of the error specific to *Category-I* and *Category-II* is 2.42% and 3.71% respectively as indicated in Table 8.6 (third and second last row).

8.6.2.2 Type of substitution errors

A detailed substitution error analysis was conducted for *Konkani_test_dataset* for *Category-I* and *Category-II* fonts. The substitution errors are divided into six types based on character/syllable¹⁰ that generated errors. First four are based on the four types of Devanagari character set and the other two types of error are based on Devanagari syllables¹⁰ and Numbers. They are named as Devanagari signs¹¹ and other characters (E1), Devanagari Dependent vowels¹¹ (E2), Devanagari consonants¹¹ (E3), Independent vowels¹¹ (E4), Devanagari Syllables¹⁰/Conjunct consonant (E5) and Numbers (E6). Table 8.7 shows the average percentage of error exhibited by *Konkani_test_dataset* for *Category-I* and *Category-II* fonts for each type of the substitution error.

Additional details of these substitution errors are provided in Table C.2 and Table C.3 (refer Appendix C for *Category-I* and *Category-II* respectively). Examples for each of the errors is

¹⁰<https://w3c.github.io/ilreq/>

¹¹<https://www.unicode.org/charts/PDF/U0900.pdf>

Font category	<i>Konkani_test_dataset</i> with font	% of errors			
		insertion	substitution	deletion	CER
<i>Category -I</i>	<i>Amiko</i>	0.34%	0.22%	1.53%	2.09%
	<i>Lohit Devanagari</i>	0.34%	0.22%	1.53%	2.09%
	<i>Biryani</i>	0.34%	0.22%	1.53%	2.09%
	<i>Nakula</i>	0.32%	0.35%	1.40%	2.07%
	<i>Cambay</i>	0.27%	0.27%	1.81%	2.35%
	<i>Eczar</i>	0.77%	0.48%	1.67%	2.92%
	<i>Gargi</i>	0.38%	0.30%	1.84%	2.52%
	<i>Asar</i>	0.49%	0.39%	2.31%	3.19%
<i>Category -II</i>	<i>Baloo</i>	1.38%	0.71%	1.49%	3.58%
	<i>Aparajita</i>	0.76%	1.60%	1.47%	3.83%
<i>Category -I</i> average		0.41%	0.31%	1.70%	2.42%
<i>Category -II</i> average		1.07%	1.16%	1.48%	3.71 %
Overall average		0.54%	0.48%	1.66%	2.67%

Table 8.6: Percentage of insertion, substitution and deletion errors exhibited by *Konkani_Multi_Font* OCR

Error type name	Substitution error type	<i>Category-I</i>	<i>Category-II</i>
E1	Devanagari signs and other characters	0.102%	0.063%
E2	Devanagari Dependent vowels	0.077%	0.124%
E3	Devanagari consonants	0.066%	0.681%
E4	Independent vowels	0.044%	0.062%
E5	Syllables/Conjunct consonants	0.017%	0.031%
E6	Numbers	0.002%	0.012%

Table 8.7: Average percentage of errors exhibited by *Konkani_test_dataset* for *Category-I* and *Category-II* fonts for each type of substitution error

shown in Table C.4 (refer Appendix C). These words were top 38 words incorrectly recognized by *Konkani_Multi_Font* OCR, always giving zero percent accuracy. The details of each of these substitution type errors along with percentage and example are here:

- (E1): Devanagari signs and other characters:** this type of substitution errors are mainly introduced by Devanagari signs¹¹ and other character such as single quotation mark, question mark, colon, round bracket, etc. Through Table 8.7, it is observed that *Category-I* shows 0.102% and *Category-II* shows 0.063% of overall average errors respectively. As a result, the dataset that uses the *Category-II* font contains less errors of this kind. In both categories these errors are mostly due to incorrect recognition of character ‘ (left single quotation mark) and ’ (right single quotation mark). And partial errors have occurred due to Devanagari sign *Anusvara* ँ (U+0902)¹¹ and *Candrabindu* ँ (U+0901)¹¹ (refer Table C.2 and Table C.3 (refer Appendix C) for more details). Some of the examples that contributed for incorrect recognition of Devanagari sign ँ *Anusvara* is primarily due to the formation of complex syllables, e.g., word अर्दात [əɾəḍāṭ] was recognized as अर्दात [əɾəḍat], to identify *Anusvara* in this word is a challenging task (for more examples refer Appendix C Table C.4 Sr.No 1-5). Errors exhibited by a single quotation are due to words such as कुंभार’, which was recognized as कुंभार’ These two are different quotation marks. For more examples refer Appendix C Table C.4 (Sr.No 29-38).
- (E2): Devanagari Dependent vowels:** This type of substitution errors are due to incorrect recognition of Devanagari dependent vowels¹¹. It is observed that *Category-I* and *Category-II* showed 0.077% and 0.124% of total errors respectively. In both category these are primarily due to the dependent vowel ॅ (U+0945) *Candra E*¹¹ (refer Appendix C Table C.2 and Table C.3 for more examples). Here *Category-II* showed a higher percentage of this error compared to *Category-I* fonts.
- (E3): Devanagari consonants:** This type of error is based on incorrect recognition of Devanagari consonant¹¹. The *Category-I* showed 0.066% of errors. Mostly due to consonant ऋ, which is recognized as ॠ and ॡ is recognized as ॢ. refer to Table C.2 (refer Appendix C) for more examples.
- The *Category-II* showed 0.681% of errors. Mostly, this is shown by consonant ॢ, which is often recognized as ॣ. This is due to its appearance exhibited by font *Aparajita* and *Baloo*, e.g. हँवे ([hãvẽ] gloss: I did) (see Table 8.1). More example can be referred in Table C.3 (refer Appendix C). In comparison to *Category-I*, *Category-II* displayed a higher percentage of consonant type of error.
- (E4): Independent vowels:** This type of error encompasses the incorrect recognition of De-

vanagari vowels¹¹. It is observed that *Category-I* and *Category-II* contribute towards errors 0.044% and 0.060% of average errors for each category, respectively. Mostly due to incorrect recognition of independent vowels such as ऐ (U+090d)¹¹. Which is mostly incorrect recognition of Devanagari dependent vowel ँ (Candra E). It is important to note that its occurrence is mostly seen in the foreign language borrowed words. e.g. ऐन्थनी (noun: Anthony), ऐड ([ɛd] gloss: Add), ऐपल ([ɛpəl] noun: Apple), etc. As an exception in Konkani orthographical rules (Bhave et al., 2015). Examples of an error that occurred due to incorrect recognition of ँ (Candra E) are word ऐन्थनी [ɛntʰəni] is recognized as ऐन्थनी [ɛntʰəni] (refer Table C.4 Sr.No 6-21 (refer Appendix C)). It is observed that these words are borrowed words from foreign languages like ऐन्थनी (noun: Anthony), ऐपल (gloss: Apple), ऐड (gloss: And), etc. And the word Anthony is written as ऐन्थनी in Hindi due to its pronunciation. It is also important to note that these words were less frequent in the *Konkani_Train_dataset*. Thus, errors are generated due to the same.

(E5): Syllables/Conjunct consonants: This type of error is incorrect recognition of Devanagari syllable¹⁰/Conjunct consonants. The *Category-I* has 0.017% of error. These mostly due to incorrect recognition of conjunct consonant/syllable द [d̪d̪ə] , which is recognized as ह [hə] . And other example is ड [d̪d̪ə] , which is recognized as ड [d̪u].

The *Category-II* has 0.031% of total errors The examples can be referred in Table C.3 (refer Appendix C). The errors exhibited by syllables/conjunct consonants appear in Table C.4 Sr.No 22-28 (refer Appendix C). For example, word एकगडो [ekəgəʈʰə] was recognized as एकगडो [ekəgəʈʰə].

(E6): Numbers: Errors occurred due to incorrect recognition of numbers. Here, *Category-I* and *Category-II* showed 0.002% and 0.012% of overall overage errors respectively. For example such as number 9 was recognized as ५ [bʰə], number 27 recognized as ३१ [ʃa], and number 37 is recognized as ३ [ə]. However, this error percentage is negligible.

The top 50 words having frequent occurrences, which are always correctly recognized for all ten fonts, are shown in Table C.1 (refer Appendix C). Overall, it is observed the model faces challenges to recognize ‘ (left single quotation mark), ’ (right single quotation mark) and ँ (Candra E). These errors can be handled by post-processing error correction approach.

8.6.3 Comparison of top 10 substitution errors exhibited by Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *Category-I* fonts

#	Devanagari Tesseract OCR		<i>Konkani_Multi_Font</i> OCR	
	Actual character	Error %	Actual character	Error %
1	र Consonant [rə]	0.112%	'(U+2019)(Right Single Quotation mark)	0.045%
2	'(U+2019)(Right Single Quotation mark)	0.101%	ँ Independent vowel	0.04%
3	ल Consonant	0.057%	'(U+2018) (Left Single Quotation mark)	0.037%
4	'(U+2018) (Left Single Quotation mark)	0.055%	ँ(U+0945) <i>Candra E</i>	0.03%
5	“ Left double quotation mark	0.053%	ँ Independent vowel	0.018%
6	? Question mark	0.039%	व Consonant	0.01%
7	ँ(U+0945) <i>Candra E</i>	0.034%	न Consonant	0.009%
8	व Consonant	0.024%	ँ Independent vowel	0.008%
9	ँ Independent vowel	0.017%	ं <i>Anusvara</i>	0.008%
10	े Devanagari vowel sign	0.008%	ल Consonant	0.008%

Table 8.8: Top 10 substitution errors exhibited by Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *Category-I* fonts

Table 8.8 shows the top 10 errors exhibited by Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *category-I* fonts. It is observed that Devanagari Tesseract OCR exhibited three most common substitution errors were associated with the consonant र [rə], ' right single quotation mark, and third was by consonant ल [lə].

Konkani_Multi_Font OCR exhibit errors top 3 errors are the '(U+2019) (Right single quotation mark), ँ(U+0945) (*Candra E*), and '(U+2018) (left single quotation mark). It is observed that the consonant errors are less compared to Devanagari Tesseract OCR. It is much easy to handle errors due to Devanagari sign than consonant.

8.6.4 Performance of *Konkani_Multi_Font* OCR on real data

The output of Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR output on real data newspaper clip can be seen in Figure 8.3. The words marked in red color indicate incorrectly recognized words. Some examples of these OCR outputs are given in Table 8.9. The Konkani words पातळ्ळां [paṭṭəllā], मेळ्ळी [meḷli] and फायद्यांत [fa.jəḍjāṭ] are correctly recognized by *Konkani_Multi_Font* OCR and not by Devanagari Tesseract OCR. The word मॅरोथॉन ([məroθhən] gloss: marathon) is a foreign language borrowed word, and overlapping of characters can be seen in the input image. Thus, these words are challenging to recognize correctly by Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR.

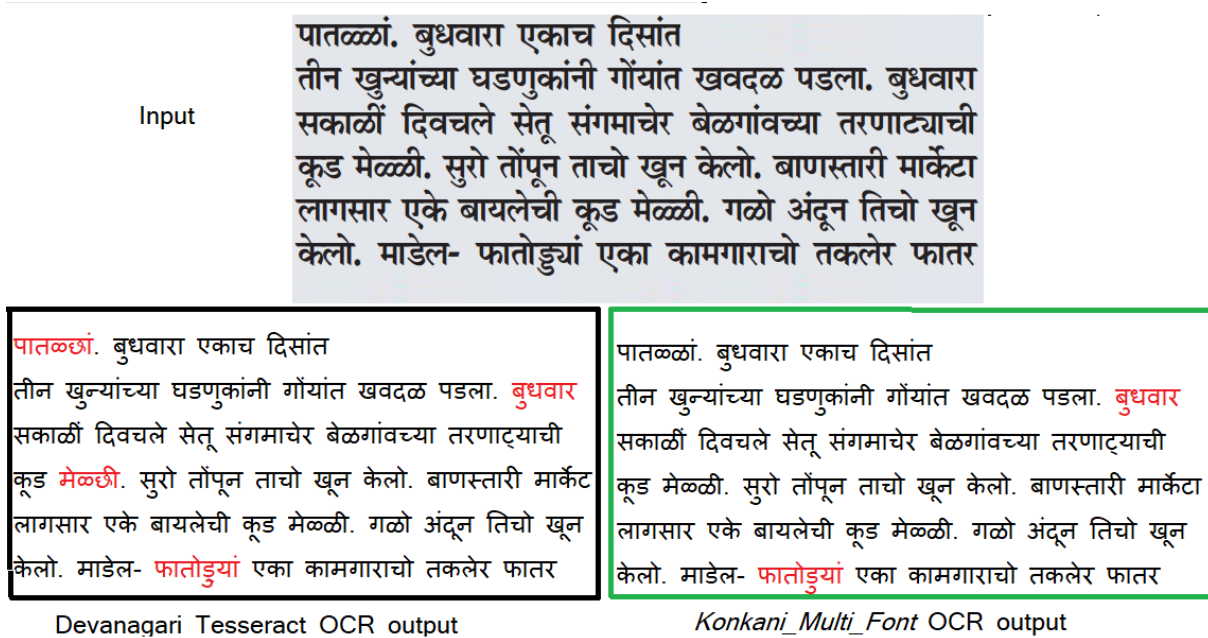


Figure 8.3: Output of Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR on newspaper clip

The Konkani multi-font OCR has also been tested with a Konkani text in two-column book format on ten pages having 5963 words, which exhibits 98.80% word accuracy. Sample input image is shown in Figure 8.4 along with its *Konkani_Multi_Font* OCR output. The incorrect characters are indicated in red. It is also tested with 40 Konkani articles from *Bhaangarbhuin* Konkani newspaper¹², which exhibits 97.35% word accuracy. Figure 8.5 shows the sample Input image and OCR output. This accuracy is higher than the average accuracy on *Konkani_test_dataset*.

¹²<https://epaper.thegoan.net/>

Input	Devanagari Tesseract OCR output	<i>Konkani_Multi_Font</i> OCR output
पातळ्ळां	पातळ्ळां	पातळ्ळां
मेळ्ळी	मेळ्ळी	मेळ्ळी
फायद्यांत	फायद्यांत	फायद्यांत
बिलां	बिला	बिलां
मुडी	मुडी	मुडी
फातोड्ड्यां	फातोड्ड्यां	फातोड्ड्यां
मैराथॉन	मराथॉन	मंरथोन
सर्त	स्त	सत
गोंयांत	गोंयांत	गोंयांत

Table 8.9: Words from newspaper clip in different fonts and its corresponding Devanagari Tesseract OCR and *Konkani_Multi_Font* OCR output

The reason for same is this data is Konkani data with minimum foreign words.

8.7 Konkani OCR interface

OCR for Konkani language has been developed and made available in Goa University private network for testing p. The OCR tool takes an image as input, and output is OCR recognized Devanagari text as shown in Figure 8.6.

8.8 Conclusion

This research addressed the need for a dedicated OCR for the Konkani language, a low-resource language officially written in the Devanagari script. Thus, this work has developed a *Konkani_Multi_Font* OCR using Tesseract LSTM. On evaluating the existing Devanagari Tesseract OCR performance on a diverse range of fonts using the *Konkani_test_dataset* the average word accuracy was 92.43%. *Konkani_Multi_Font* OCR gave an average word accuracy of 95.94% on *Konkani_test_dataset*. In practical applications, the *Konkani_Multi_Font* OCR showed remarkable word accuracy of 98.80% on Konkani books and 97.35% on Konkani newspaper data by increasing its accuracy for

थंय आशिल्ल्या यामुनाचार्याक गाळी घाल्यो आनी कर घेवन रोखडोच कोलाहलाकडेन येवपाक सांगलें. यामुनाचार्याक ताची तिडक आयली. ताणें म्हळें 'म्हजो गुरू कर भरचो ना. ताचेबदला तुज्या गुरूकडेन हांव शास्त्रार्थ करपाक तयार आसां'. हें आयकून शिष्याक तिडक आयली आनी ताणें परत वचून कोलाहलाक घडिल्ली खबर सांगली.

कोलाहलाकूय हें सगळें आयतकच बारा-पंदरा वर्सांच्या यामुनाचार्याची तिडक आयली. ताणें राजाक ताका आपोवणें धाडपाक सांगलें.

दुतांकडल्यान जेन्ना राजाक यामुनाचार्य वादविवादाक तयार आसा म्हणपाचें कळ्ळें, तेन्ना राजान ताका दरबारांत येवपाखातीर पालखी धाडली.

कवी म्हणून ताच्या विकासाचे दोन पांवडे दिसतात. पयलो १९०० वर्सा मेरेनचो आनी दुसरो ते उपरांतचो. ताच्या पयल्या काळखंडांतले कवितेचेर एडमंड स्पेंन्सर, शॅली, प्री रॅफेएलायड्स हांचो प्रभाव दिसता. पूण ताचे उपरांतचे कवितेंत परिपक्वताय आनी समृद्धतायेचो एक वेगळोच अणभव येता. यिट्साच्या काव्यग्रंथांत द विंड अॅमॉंग द रीड्स (१८९९), द शॅडोई वॉटर्स (१९००), इन द सॅव्हन वूड्स (१९०३), द ग्रीन हॅलमॅट अँड अदर पोयम्स (१९१०), रिस्पॉन्सिबिलिटीस (१९१४), द टावर (१९२८), द वायडिंग अँड अदर पोयम्स (१९२९) आनी लास्ट पोयम्स (१९४०) हांचो आस्पाव आसा. प्रतीकवादी प्रवृत्तीचो जाणविकायेन आविश्कार आपल्या कवितांतल्यान घडोवपी इंग्लिशोतल्या सुरवातीच्या कवींभितरलो यिट्स

थंय आशिल्ल्या यामुनाचार्याक गाळी घाल्यो आनी कर घेवन रोखडोच कोलाहलाकडेन येवपाक सांगलें. यामुनाचार्याक ताची तिडक आयली. ताणें म्हळें 'म्हजो गुरू कर भरचो ना. ताचेबदला तुज्या गुरूकडेन हांव शास्त्रार्थ करपाक तयार आसां'. हें आयकून शिष्याक तिडक आयली आनी ताणें परत वचून कोलाहलाक घडिल्ली खबर सांगली. कोलाहलाकूय हें सगळें आयतकच बारा-पंदरा वर्सांच्या यामुनाचार्याची तिडक आयली. ताणें राजाक ताका आपोवणें धाडपाक सांगलें.

दुतांकडल्यान जेन्ना राजाक यामुनाचार्य वादविवादाक तयार आसा म्हणपाचें कळ्ळें, तेन्ना राजान ताका दरबारांत येवपाखातीर पालखी धाडली.

कवी म्हणून ताच्या विकासाचे दोन पांवडे दिसतात. पयलो १९०० वर्सा मेरेनचो आनी दुसरो ते उपरांतचो. ताच्या पयल्या काळखंडांतले कवितेचेर एडमंड स्पेंन्सर, शॅली, प्री रॅफेएलायड्स हांचो प्रभाव दिसता. पूण ताचे उपरांतचे कवितेंत परिपक्वताय आनी समृद्धतायेचो एक वेगळोच अणभव येता. यिट्साच्या काव्यग्रंथांत द विंड अॅमॉंग द रीड्स (१८९९), द शॅडोई वॉटर्स (१९००), इन द सॅव्हन वूड्स (१९०३), द ग्रीन हॅलमॅट अँड अदर पोयम्स (१९१०), रिस्पॉन्सिबिलिटीस (१९१४), द टावर (१९२८), द वायडिंग अँड अदर पोयम्स (१९२९) आनी लास्ट पोयम्स (१९४०) हांचो आस्पाव आसा. प्रतीकवादी प्रवृत्तीचो जाणविकायेन आविश्कार आपल्या कवितांतल्यान घडोवपी इंग्लिशोतल्या सुरवातीच्या कवींभितरलो यिट्स

(a) Input from Konkani text book

(b) Konkani_Multi_Font OCR output

Figure 8.4: Input Konkani book and Konkani_Multi_Font OCR output

Konkani documents and minimizing Devanagari consonant errors. This development holds great promise for accelerating the creation of a Konkani language corpus and digitizing Konkani books and old newspaper articles. This will serve as a base for other NLP applications for Konkani.

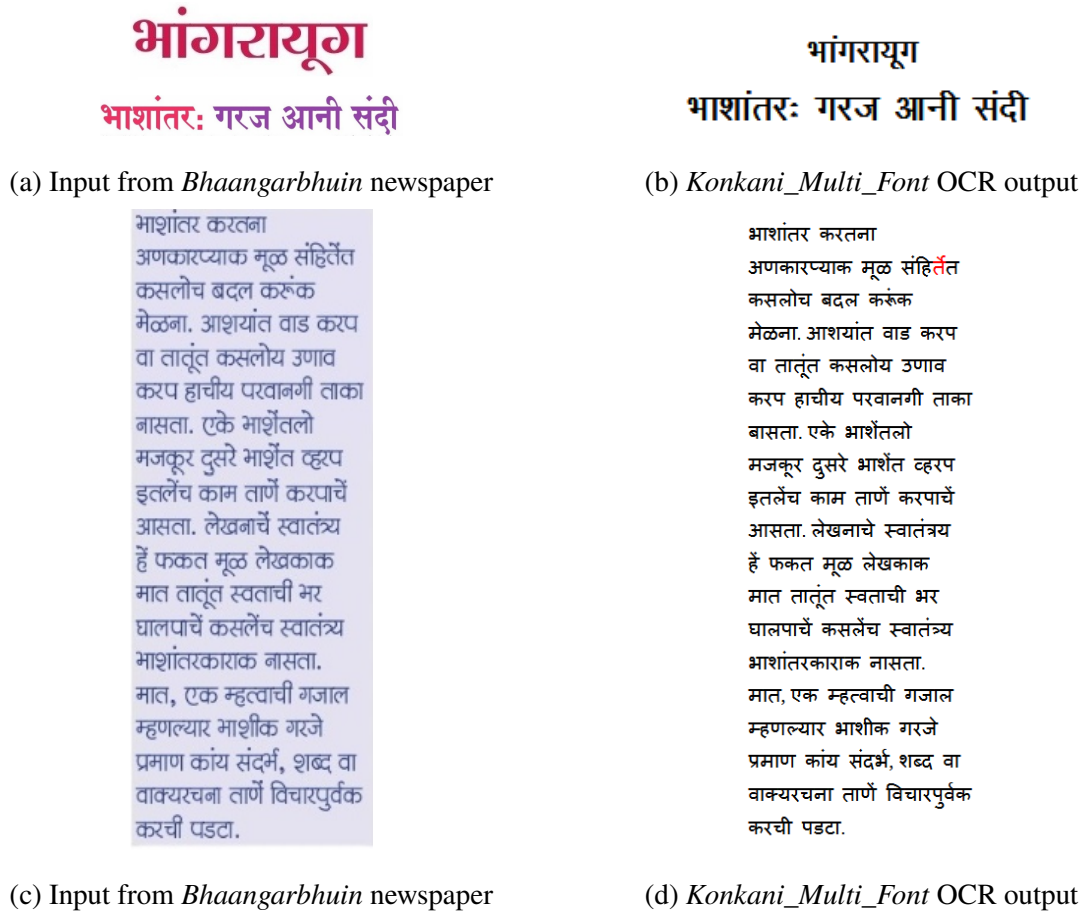


Figure 8.5: Input *Bhaangarbhuin* newspaper clip and *Konkani_Multi_Font* OCR output

OCR for konkani

Upload image to transcribe

Choose File

book.PNG

Submit

Uploaded image

यिद्दीश भास आनी साहित्य

उपरांत मध्य युरोपांतल्यान उदेंत युरोपांत चड जावंक लागलो. चड करून ही भास रशिया आनी रुमानियेंतले ज्यू लोक उलयताले.

यिद्दीश भाशेचो उगम आनी इतिहास हो ज्यू लोकांच्या धर्म, इतिहास आनी संस्कृताय हांचेवांगडा बांदिल्लो आसा. जर्मनींतल्या न्हायनलँड वाठारांत, इकराव्या - बाराव्या शेंकड्यांत ज्यू भाशेचो जल्म जालो आसुंये. तशेंच सेमिटिक संस्कृतायेंतलीं उतरां आनी हिब्रू लिपी ह्या दोन मुखेल घटकांच्या आदाराचेर हे भाशेचो विकास जालो.

पुर्विल्ली यिद्दीश (१५०० मेरेन) : व्होर्म्सच्या माखसोरांत

Extracted text

Download Text

यिद्दीश भास आनी साहित्य

उपरांत मध्य युरोपांतल्यान उदेंत युरोपांत चड जावंक लागलो. चड करून ही भास रशिया आनी रुमानियेंतले ज्यू लोक उलयताले.

यिद्दीश भाशेचो उगम आनी इतिहास हो ज्यू लोकांच्या धर्म, इतिहास आनी संस्कृताय हांचेवांगडा बांदिल्लो आसा. जर्मनींतल्या न्हायनलँड वाठारांत, इकराव्या - बाराव्या शेंकड्यांत ज्यू भाशेचो जल्म जालो आसुंये. तशेंच सेमिटिक संस्कृतायेंतलीं उतरां आनी हिब्रू लिपी ह्या दोन मुखेल घटकांच्या आदाराचेर हे भाशेचो विकास जालो.

पुर्विल्ली यिद्दीश (१५०० मेरेन) : व्होर्म्सच्या माखसोरांत

Figure 8.6: Konkani OCR tool

Chapter 9

Conclusion

This research investigates the variability in the usage of Devanagari characters, ligatures, and syllables (akshara) across different languages and fonts. The performance of existing Optical Character Recognition (OCR) systems is inconsistent when applied to various fonts. The implemented Weighted Syllable based Spell Correction (WSSC) approach addresses substitution errors for two different fonts *Aparajita* and *Nakula*. Additionally, the error correction strategy using MLM-BERT suggests words based on contextual information, while the *MLM-BERT with Norvig spell corrector (MB-NSC)* approach suggests words based on contextual information as well as the nearest likely word. Two automatic dataset creation procedures were devised, resulting in creation of OCR and post-processing datasets for Hindi, Konkani, Marathi, and Nepali. This work created a Devanagari multilingual error detection model utilizing mBERT and CNN, supporting four languages: Hindi, Konkani, Marathi, and Nepali. A multi-font Hindi text recognizer that supports five fonts was created, which showed negligible deletion errors. An OCR system supporting ten fonts was developed for Konkani. This Konkani OCR has been created using Tesseract Devanagari OCR model and made available for testing purposes within the Goa University private network, providing a foundational resource for future Natural Language Processing (NLP) applications. This work underscores the significance of error analysis in research and has led to contributions.

9.1 Contributions

1. **Empirical study of Devanagari character, *ligatures* and syllables in Konkani, Hindi, Marathi and Nepali.**

Devanagari has numerous characters, vowels, consonants, *ligatures* and syllables. Stud-

ied the similarities and differences of these characters, *ligatures* and syllables among four languages written using Devanagari; Konkani, Hindi, Marathi and Nepali. The binary Jaccard similarity score computation was calculated for each language pair. It is observed that for the Konkani-Marathi pair, the binary Jaccard similarity is 0.41, which is the highest. Hindi resembles more closely to Marathi with a binary Jaccard score of 0.40, while Nepali displays a closer resemblance to Hindi, with a score of 0.31.

2. Datasets Created for OCR and OCR post-processing performance study.

Three procedure were proposed to create the OCR and OCR post processing dataset viz, (i) Manual procedure to create OCR dataset, (ii) *text2img-doc* procedure to create OCR dataset and (iii) *post-OCR-align* procedure to create post processing dataset. This can be used to create data for any languages having varying font and font size. Using these procedures the following dataset were created for Konkani, Hindi, Marathi and Nepali as listed below.

(a) Hindi datasets for Aparajita and Nakula fonts

- *BM_Aparajita* test dataset
- *BM_Nakula* test dataset

(b) Hindi datasets for OCR post processing.

- *Hin_test_dataset*
- *Hindi-IIITH-dataset*

(c) Konkani datasets for OCR and OCR post-processing.

- *Konkani_OCR_large_test_dataset*
- *Konkani_OCR_subset_test_dataset*
- *Konkani_EC_test_dataset*

(d) Konkani dataset for Multi-Font to test OCR (*Konkani_test_dataset*).

(e) Indic datasets for OCR and OCR post-processing

- *Indic-OCR-test-dataset*
- *Indic-post-OCR-dataset* (to train and test OCR post-processing models)

3. Algorithms Proposed and Implemented for OCR Error Correction.

Three error correction approaches are proposed as listed below:

- (a) Error correction with Weighted Syllable based Spell Correction (WSSC): This handles substitution errors and it is robust.
 - (b) Error correction using Masked-Language Modeling with Bidirectional Encoder Representations from Transformers (MLM-BERT) : This provides suggestion based on context. It showed improvement in word accuracy on Hindi dataset. This showed its effectiveness for inflectional languages by suggesting word based on context.
 - (c) Error correction with the hybrid approach: MLM-BERT and Norvig Spell Corrector (MB-NSC): This provide contextual suggestion as well as suggestion with nearest likely word based on edit distance. And showed improvement in word and character accuracy on Konkani documents.
4. **Algorithm Proposed and Implemented for OCR error detection: Konkani, Hindi, Marathi, Nepali using Multilingual BERT (mBERT).**: Error detection approach has been proposed using mBERT and CNN. Five error detection models were created; four for individual languages Konkani, Hindi, Marathi and Nepali and fifth one was Devanagari multilingual model supporting these four languages. The Devanagari multilingual error detection model build showed similar F-score to that for individual language.
5. **Multi-font text recognizer for Hindi (MDF_OCR).** This work studies the percentage of type of errors of existing OCR on different fonts *Nakula*, *Baloo*, *Dekko*, *Briyani* and *Aparajita*. It was observed that types of OCR errors are not consistent for different fonts. This work has contributed to the design of multi-font OCR for Hindi, supporting five fonts using LSTM (open-source OCRopus), which showed negligible deletion errors.
6. **Multi-font OCR for Konkani (Konkani_Multi_Font OCR):** Konkani OCR is created using the base model Devanagari Tesseract OCR model. An OCR for Konkani which is available within Goa University (URL: <http://192.168.101.53:3000/>).

9.2 Summary

The Devanagari script is composed of numerous letters which consist of characters, syllables, and *ligatures* (consonant cluster). No study has been done to investigate the usage of these letters: characters, *ligatures*, and syllables (*ligatures* with dependent vowels and sign) in lan-

guages. Also, the *ligatures* appears differently for various fonts. Thus, It is essential to study the similarities and differences of characters, syllables, and *ligatures* of every language. The study was conducted on languages viz., Konkani, Hindi, Marathi and Nepali. Upon investigating, it was found that there are certain characters and *ligatures* specifically utilized by each language differently. The binary Jaccard similarity score was calculated for each language pair. It is observed that for the Konkani-Marathi pair, the binary Jaccard similarity is 0.41, which is the highest. Hindi resembles more closely to Marathi with a binary Jaccard score of 0.40, while Nepali displays a closer resemblance to Hindi, with a score of 0.31. However, it is crucial to note that dissimilarities, as suggested by the binary Jaccard similarity score, signify that each language possesses unique syllables and *ligatures*.

Thus, usage of Devanagari characters, *ligatures* and syllables varies for each of the language and its appearance differs for different fonts. It is essential to study the performance of the existing OCR systems for Devanagari script with text having different font. But there is no multi-font Devanagari OCR was available for languages. The available Hindi OCR test dataset is in single font. This work proposes a procedure to create OCR dataset and further create the Hindi multi-font datasets with *Aparajita* and *Nakula* fonts (i.e. *BM_Aparajita*, *BM_Nakula* dataset). The corpus selected to create these datasets was identical to that used in testing the Multilingual Indic OCR. The corpus selected to create these datasets was identical to that used in testing the Multilingual Indic OCR. And studied the performance of the existing OCRs on these datasets. Performance study of existing OCRs were conducted on these datasets. It is observed that the OCR accuracy varies for different fonts. This work proceeds to correct errors generated by these two OCR'd texts. Upon analyzing OCR'd text, errors are categorized into three types, viz., insertion, deletion, and substitution. Substitution errors and their frequencies are stored in a frequency-based error mapping tuple. Thus, a Weighted Syllable Spell Correction (WSSC) approach was proposed for handling substitution errors. It was tested on OCR'd output (Tesseract, Indsenz and e-Aksharayan OCR output) with datasets having *Aparajita* and *Nakula* fonts (i.e. *BM_Aparajita*, *BM_Nakula* dataset). This approach detects errors using a dictionary and suggests the appropriate word for each incorrect one using WSSC approach. WSSC recommends the appropriate syllables based on the frequency-based error mapping tuple and suggests a likely word. Result shows that accuracy of Tesseract OCR output for *BM_Aparajita* dataset increased from 65.95% to 67.19%. Similarly, the Indsenz accuracy on *BM_Aparajita* dataset increased from 53.23% to 55.00%. A small marginal increased in the OCRs accuracy for *BM_Nakula*

dataset was observed. It is observed that there is improvement in substitution errors for OCRd text in both fonts. However, this approach does not handle insertion and deletion errors. Moreover, the new syllable substitution errors generated through text in different fonts can be handled by updating the frequency-based error mapping tuple, thereby demonstrating its adaptability.

Handling the insertion and deletion errors is difficult. Other challenges occurred during post processing is the Indian language are highly inflectional. Thus the motivation to provide suggestion for the incorrect word based on context. The proposed approach was error correction using MLM-BERT. This requires OCR post processing dataset to test the approach. Further *text2img-doc* and *post-OCR-align* procedure to create OCR and post processing dataset is devised. And *Hindi_test_dataset* and *Hindi-IIITH-dataset* for Hindi is created. Error correction using the MLM-BERT approach was tested on *Hindi_test_dataset* and *Hindi-IIITH-dataset*. It gives contextual suggestions for the masked erroneous word, which showed an improvement of 3.58% word accuracy for Hindi. This approach performs well by providing contextual suggestions for incorrect word. It was also observed that in some cases, the model does not provide any suggestions. These discrepancies may arise due to incorrect consecutive words or contextual considerations. Another observation is that while the suggested word may be contextually appropriate, it might still be incorrect. In some cases, the length of the suggested word can be significantly greater or less than the actual word, which affects the character accuracy. Instead of reducing insertion errors, it increases them.

To achieve the main objective of this research namely an OCR solution for Konkani documents was proposed . This OCR solution required Konkani dataset. Accordingly, dataset for Konkani i.e. *Konkani_OCR_large_test_dataset* (having 3.6 million words), *Konkani_OCR_subset_test_dataset* and *Konkani_EC_dataset* has been created. It explores Tesseract Hindi OCR on Konkani documents (*Konkani_OCR_large_test_dataset*) having 3.6 millions words, yielding a word accuracy of 89.25%. The evaluation revealed a CER of 4.15% comprising of insertion errors (1.02%), substitution errors (2.09%), and deletion errors (1.04%). The top most error was primarily associated with the syllable ॐ ([ə]). Therefore, the utilization of Hindi OCR for Konkani text has revealed specific challenges in terms of performance. This understanding significantly contributed to development and subsequent improvement of the accuracy of Konkani OCRd text. This was achieved through the proposed MB-NSC error correction approach, i.e., combining the state-of-the-art Masked Language Modeling BERT and Peter Norvig spell corrector. This

approach has been tested on *Konkani_EC_dataset* dataset. The MB-NSC approach for Konkani identifies contextual information to provide precise word corrections, effectively handling words with special characters unique to the Konkani language, including those containing the character ँ ([ə]). Thus, accomplishing a 2.74% enhancement in word accuracy and 2.60% of character accuracy compared to the current Tesseract Hindi OCR and demonstrating its efficacy in rectifying errors in the OCR output text during post-processing. However, it could not handle words with Zero Width Joiner. This can be addressed by developing a OCR specific to Konkani.

Further two dataset *Indic-OCR-test-dataset* and *Indic-post-OCR-dataset* have been created supporting four languages; Hindi, Konkani, Marathi and Nepali. *Indic-OCR-test-dataset* was used to study the performance of two Hindi Devanagari OCRs, namely Tesseract and Indsenz on these four languages. The findings indicate that Tesseract OCR outperforms Indsenz OCR across all four languages. Consequently, Tesseract OCR was selected as the basis for creating the training and test dataset called *Indic-post-OCR-dataset*. Subsequently *Dev-multi-lingual-Error-Detection-Model* using the *Indic-post-OCR-dataset* and mBERT with CNN techniques has been created. The model's performance was evaluated on the four Indian languages mentioned above. Additionally, separate models are created for each language. The test results reveal that the F-score of the language-dependent models is similar to that of the *Dev-multi-lingual-Error-Detection-Model*. The study demonstrates the potential of leveraging existing tools designed for resource-rich languages to provide solutions for low-resource languages. Accordingly, a model was successfully developed that supports all four languages. Error analysis indicates that the model accurately identifies real errors but struggles with classifying interjections, numbers and code-mixed sentences. Furthermore, the model faces challenges in detecting words from different domains, particularly English words that have been transliterated. Addressing these shortcomings requires further investigation.

It is clear from this result that language specific multi-font OCR is essential. This initiated, the study of the effect of multiple fonts on Hindi OCRs. It was observed that the percentage of various error types differs across different fonts. Thus, a multi-font OCR (MFD_OCR) using LSTM (open-source OCRopus) and Hindi dataset was built. which demonstrated consistent character accuracy across all test datasets and achieved comparatively good accuracy for text in the *Dekko* and *Biryani* fonts. Detailed error analysis showed that, unlike other Hindi OCRs, the insertion and deletion errors for the multi-font Hindi OCR (MFD_OCR) were consistent across

all test datasets for each font. Deletion errors were negligible, i.e., less than 1.4%. Additionally, Tesseract OCR had the lowest substitution error rate compared to all other OCRs and achieved the highest accuracy for text in the *Nakula*, *Baloo*, and *Aparajita* fonts.

Through this investigation, it is clear that there is a need for a dedicated OCR for Konkani supporting multi-font. Konkani multi-font data was created (*Konkani_test_dataset*) supporting 10 fonts. Initially the performance of existing Devanagari Tesseract OCR on *Konkani_test_dataset* was investigated. It was observed that Devanagari Tesseract OCR accuracy varies for different fonts. Moreover, the topmost error was caused by a consonant. This work lead to development of *Konkani_Multi_Font* OCR using Devanagari Tesseract OCR as base model. On evaluating the existing Devanagari Tesseract OCR performance on a diverse range of fonts using the *Konkani_test_dataset* the average word accuracy was 92.43%. *Konkani_Multi_Font* OCR gave an average word accuracy of 95.94% on *Konkani_test_dataset*. In practical applications, the *Konkani_Multi_Font* OCR showed word accuracy of 98.80% on Konkani books and 97.35% on Konkani newspaper data. The comprehensive error analysis presented in the study provides valuable insights for further work in post-processing error correction approach. Konkani OCR is hosted and available on Goa University private network for testing purpose. It serve as a tool for creating resources for other NLP applications in Konkani.

9.3 Scope for Future Research

The following work can be carried out in future work;

1. **Expansion of Multi-Font OCR for Handwritten Text Recognition:**

Investigate and implement techniques to enhance the capability of the Multi-Font OCR system to accurately recognize handwritten text.

2. **Error Correction for all the languages using Devanagari:**

Develop algorithms and models specifically tailored for error correction in Devanagari. Utilize linguistic and contextual features to improve the accuracy of error correction for Devanagari text.

3. **Exploring advanced BERT Models for Error Correction:**

Incorporate state-of-the-art BERT (Bidirectional Encoder Representations from Trans-

formers) models with enhanced capabilities for error correction and detection. Fine-tune BERT models on domain-specific data to improve performance on error correction tasks.

4. Error Detection and Correction for Numbers:

Design and implement algorithms for the detection and correction of errors in numerical data, including digits, numerical expressions, and numerical identifiers.

5. Error Detection and Correction for foreign words appearing in Indian languages:

Design and implement algorithms for the detection and correction of errors foreign words.

6. Error Detection and Correction for Indian language code-mixed data:

Design and implement algorithms for the detection and correction of errors in code-mixed data of Indian languages.

Appendix A

Examples of Devanagari characters, *ligatures* and syllables:

Konkani, Hindi, Marathi and Nepali

A.1 Independent vowels

Char acter	Konkani			Hindi			Marathi			Nepali		
	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss
अ	अशें	asheM	like that	अपने	apane	Ours	असे	ase	like that	अनुसार	anusAra	According to
आ	आनी	AnI	And	आज	Aja	Today	आणि	ANi	And	आफ्नो	Aphno	his
इ	इतलो	italo	So much	इस	isa	This	इथे	ithe	Here	पाइन्छ	pAinCha	found

Char acter	Konkani			Hindi			Marathi			Nepali		
	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss
ई	आई	AI	Mother	गई	gaI	Went	होईल	holla	will be done	दुई	duI	Two
उ	उपरांत	uparAMta	After	उन्होंने		They	उत्तर	uttara	Answer	उनले	unale	He
ऊ	ऊंच	UMcha	High	ऊपर	Upara	Above	घेऊन	gheUna	taking	ऊर्जा	UrjA	energy
ऋ	ऋषभ	RRiShabha	Rishabh	ऋण	RRiNa	Debt	ऋतूम	RRitU	season	ऋषि	RRiShi	sage
एँ	एँथनी	enthaniI	Anthony	-			एँकट	Ekṭa	Act	मिसिँको	misiEko	missinco
ऐ	ऐवज	aivaja	material	-			ऐवजी	AivajI	instead	ऐतिहासिक	AitihAsika	a historical
ए	एक	eka	One	लिए	lie	taken	एक	eka	One	भएको	bhaeko	Have been
ऐ	ऐकवचनी	aikavachanI	hearsay	ऐतिहासिक	aitihAsika	Historic	ऐकले	aikale	Heard	ऐनको	ainako	of the Act
ऑ	ऑस्ट्रेलिया	OsTreliyA	Australia	ऑनलाइन	OnalAina	Online	ऑगस्ट	ogasTa	August	ऑपरेशन	opareshana	operation
ओ	-			-			ओळख	ouLakha	Recognize	ओडारैमा	ouDAraimA	alarming
ओ	ओडून	oDUna	Pull	महिलाओं	mahilAoM	Women	ओळख	oLakha	identity	माओवादी	mAovAdI	Maoists
औ	औरंगजेब	auraMgajeba	Aurangzeb	औद्योगिक	audyogika	Industrial	औषध	auShadha	Medicine	औपचारिक	aupachArika	formal
अँ *	-			-			अँक्शन	Ekshana	Action	-		
*Independent vowel for Marathi												

Table A.1: Examples of Independent vowels in Konkani, Hindi, Marathi and Nepali

A.2 Consonants

Char acter	Konkani			Hindi			Marathi			Nepali		
	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss
क	ताका	tAkA	Him/Her	किया	kiyA	Done	केले	kele	Done	भएको	bhaeko	Have been
ख	खूब	khUba	A lot	लाख	lAkha	Lakh	खूप	khUpa	A lot	देखि	dekhi	From
ग	लागून	lAgUna	because	गया	gayA	Went	मग	maga	Then	लागि	lAgi	for
घ	घेवन	ghevana	Taken	घर	Ghara	House	घेऊन	gheUna	taken	घर	ghara	House
ङ	वाङ्मय	vA Nmaya	Literature	-			वाङ्निश्चय	vANanishchy	determine	जनसङ्ख्या	janasa NkhyA	Population
च	चड	chaDa	Many	चाहिए	chAhie	Do you want	तसेच	tasecha	Like that	चलचित्र	chalachitra	Movie
छ	छापून	ChApUna	Printing	कुछ	kuCha	Few	छोट्या	ChoTyA	small	छ, हुन्छ	Cha, hunCha	Six, is
ज	जाय	jAya	Need	आज	Aja	Today	म्हणजे	mhaNaje	means	जिल्लाको	jillAko	district
झ	झाडां	jhADAM	Trees	मुझे	mujhe	For Me	माझ्या	mAjhyA	My	साझा	sAjhA	Shared
ञ	ज्ञान	jnyAna	knowledge	विज्ञान	vijnAna	Science	तंत्रज्ञान	taMtrajnyAna	Technology	अञ्चलको	anchalako	of the Zone
ट	पडटा	paDaTA	Falling	राष्ट्रीय	rAShTrIya	National	भेट	bheTa	visit	एउटा	euTA	One
ठ	एकठांय	ekaThAMya	Together	बैठक	baiThaka	Meeting	ठिकाणी	ThikANI	in place	ठूलो	ThUlo	Big
ड	वांगडा	vAMGaDA	together	इंडिया	iMDiyA	India	डॉ	Ḍo	Dr	काठमाडौं	kAThamADauM	Kathmandu
ढ	गूढ	gUDha	Coop	बढावा	baDhaAvA	Promote	पुढे	pudhe	front	बढी	baDhI	more
ण	म्हूण	mhUNa	Because	पूर्ण	pUrNa	complete	आणि	ANi	And	कारण	kAraNa	cause

Char acter	Konkani			Hindi			Marathi			Nepali		
	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss
त	तो	to	He	भारत	bhArata	India	आहेत	Aheta	having	तथा	tathA	and
थ	थंय	thaMya	There	साथ	sAtha	Along with	येथे	yethe	Here	थियो	thiyo	was
द	दोन	dona	Two	देश	desha	The country	दिले	Dilē	Given	दिन	dina	Day
ध	धरून	Dharūna	Holding	प्रधान	pradhAn	Prime	अधिक	adhik	More	धेरै	dherai	A lot
न	आनी	Ani	And	करने	Karane	To do	नाही	nAhI	No	पनि	pani	also
न	पून	poon	Back	-			घेऊन	gheUn	To take	नाप्त	nApN	naapo
प	पूण	Puna	But	पर	Para	on it	आपल्या	ApalyA	Ours	रूपमा	rUpamA	as
फ	फुडें	phuDeM	Forward	खिलाफ	khilApha	Against	फक्त	Phakta	Only	आफ्नो	Aphno	his
ब	खूब	khUba	A lot	बनाने	banane	Create	बाहेर	bAhera	Out	सबैभन्दा	sabaibhandA	most
भ	भायर	bhAyara	Outside	सभी	sabhI	All	भेट	bheTa	visit	भएको	bhaeko	Have been
म	म्हाका	mhAkA	For me	में	meM	I	मी	Mi	I	नाम	nAma	name
य	ह्या	hyA	This	किया	kiyA	Did	आपल्या	ApalyA	Our	यस	yasa	This
र	कर	Kara	do	पर	Para	on it	करत	Karata	Keep doing	रहेको	raheko	exists
र	बऱ्या	baRyA	Good	-			दुसऱ्या	dusaRyA	Second	गऱ्यो	gaRyo	done
ल	लागून	Laguna	Adjacent	लिए	Lie	for you	मला	malA	For me	लागि	lAgi	for
ळ	वेळार	veLAra	On time	-			काळात	kALAta	In time	कथकळिको	kathakaLiko	Kathakaliko
ळ	नकळो	NakaLAo	Don't do it	-			हिवाळा	hivAILAA	Winter	पाळि	pAILAi	pani

Char acter	Konkani			Hindi			Marathi			Nepali		
	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss	Eg.	ITRANS	Gloss
व	हांव	hAMva	Me	विकास	vikAsa	Progress	सर्व	Sarva	All	विकास	vikAsa	development
श	अशें	asheM	like that	देश	Desha	Country.	विश्वास	vishvAsa	Faith	विशेष	visheSha	special
ष	शिक्षण	shikShaNa	Learning	वर्ष	Varsha	Year	गोष्टी	goShTI	Things	वर्ष	varSha	year
स	आसा	Asa	Having	इस	Isa	this	विकास	vikAsa	Progress	यसको	yasako	its
ह	न्हय	nhaya	River	हैं	Haim	Have	आहे	Ahe	Having	रहेको	raheko	exists

Table A.2: Examples of consonants in Konkani, Hindi, Marathi and Nepali

A.3 Dependent Vowels

Sr. No.	Devanagari Dependent vowel character	Example			
		Konkani	Hindi	Marathi	Nepali
1	ा	ना	किया	माझा	रूपमा
2	ि	कितें	विकास	आपल्या	लागि
3	ी	आनी	मंत्री	माहिती	नेपाली
4	ु	मुखार	हुए	तुम्ही	हुने
5	ू	पूण	महत्वपूर्ण	संपूर्ण	सूचना
6	ृ	संस्कृती	कृषि	कृती	पृष्ठ
7	ृ	-	-	मृत्यू	सांस्कृतिक
8	ँ	मॅच	मॅनेजमेंट	नॅशनल	कॅल्शियम
9	े	सर्वनामांचें	-	यामुळे	भने
10	े	तेन्ना	करने	आहेत	रहेको
11	ै	जैत	तैयार	पैसे	नै
12	ॉ	फुटबॉल	कॉरपोरेट	लॉकडाऊन	रॉयल
13	ो	-	-	डोळे	बसोबास
14	ो	तो	योजना	लोक	हो
15	ौ	भौशीक, गौतम	मौजूद	गौरव	काठमाडौं

Table A.3: Dependent vowel Sign examples in Konkani, Hindi, Marathi and Nepali

A.4 Some example of syllables (*ligature-vowels-signs-diacritic*) and its frequency

Sr. No.	Syllables(<i>ligature-vowels-signs-diacritic</i>)	Frequency			
		Konkani	Hindi	Marathi	Nepali
1	व्ह	10718	4	3505	0
2	ल्ल्या	18369	1	949	1

Sr. No.	Syllables(<i>ligature- vowels-signs-diacritic</i>)	Frequency			
		Konkani	Hindi	Marathi	Nepali
3	ना	14735	616	1317	1581
4	र	10272	4	383	3640
5	ल्या	94228	5227	91319	8191
6	क्त	2173	18342	14731	18166
7	र्डा	343	12	136	36
8	ज्या	7587	4886	11630	2118
9	क्यो	23	0	0	51
10	स्त	8442	17853	12853	9892
11	र्सा	5601	0	0	0
12	ट्टें	83	0	0	0
13	र्ड	1325	6355	1505	4304
14	ड्ड	409	643	21	155
15	ल्लीं	3232	1	4	0
16	ल्लङ्ग	0	0	0	2
17	ल्ला	0	0	2	4
18	कछ	0	0	0	3119
19	ट्टें	26	11	0	0
20	द्वे	44	89	231	83
21	श्च	119	214	846	683
22	ड्ड	166	8	49	14
23	ड्ड	409	643	21	155
24	श्च	872	11049	18921	17058
25	स्त्र	1056	1904	2019	4003
26	च्या	1217	9	17096	10
27	त्र्या	1390	0	767	281
28	ट्र	1453	1280	655	598
29	च्च	1501	4884	2349	3701
30	भ्र	1511	833	851	1463
31	ट्ट	1551	394	433	1997

Sr. No.	Syllables(<i>ligature- vowels-signs-diacritic</i>)	Frequency			
		Konkani	Hindi	Marathi	Nepali
32	द्ध	1589	232	7125	1125
33	द्ध	1774	9592	10597	18115
34	न्न	1774	16530	4879	17385
35	स्ट्रे	2065	917	347	757
36	ग्रे	2523	1846	511	3357
37	त्त	2554	20842	11791	10935
38	स्न	2755	3	1	854
39	त्र	10099	40127	30426	51877
40	क्ष	13864	42137	32304	20822
41	ह्या	22606	13	9381	1091
42	प्र	35289	181436	115845	115427
43	प्र	35289	181436	115845	115427
44	ल्लें	12489	2	0	0
45	व्ह	10718	4	3505	0
46	ळ्या	7939	0	4874	1
47	च्यो	7122	3	0	50
48	म्हू	6877	2	5	0
49	ड्यां	5999	0	1523	1
50	ळ्यां	5798	0	1345	0
51	र्सा	5601	0	0	0
52	ग्यां	3783	0	124	13
53	र्ती	3315	0	113	3
54	ल्लीं	3232	1	4	0
55	ड्यो	2793	0	3	113
56	ल्ल्यो	2373	0	0	2
57	श्टा	2238	1	0	173
58	ण्यां	1667	0	1082	2
59	ग्लं	1642	0	76	0
60	श्टी	1479	0	0	1

Sr. No.	Syllables(<i>ligature-vowels-signs-diacritic</i>)	Frequency			
		Konkani	Hindi	Marathi	Nepali
61	ळें	1423	0	0	0
62	त्र्या	1390	0	767	281
63	प्यां	1327	2	70	0
64	ळी	1106	0	3	0
65	ळो	1097	0	0	0
66	ळ्यो	998	0	0	0
67	न्हं	978	3	5	0
68	ल्ल्यां	884	0	109	0
69	ण्यो	878	0	0	1
70	च्यां	871	0	98	14
71	ळ्या	730	0	0	0
72	र्न	673	17	17	0
73	श्टां	668	0	0	0
74	ळे	606	0	0	0
75	र्ग	601	1	462	0
76	म्हो	582	0	16	12
77	व्हां	565	0	14	0
78	ल्लों	562	20	0	5
79	त्र्यां	539	2	1646	0
80	च्यां	500	0	6587	0
81	क्वों	499	34	5	0
82	र्द	499	0	0	0
83	श्यां	488	0	292	4
84	म्हु	474	0	4	3
85	व्हें	460	5	1445	0
86	स्टीं	459	1	11	0
87	त्यी	446	0	1	30
88	डुं	445	2	0	0
89	त्या	404	0	72	6

Sr. No.	Syllables(<i>ligature-vowels-signs-diacritic</i>)	Frequency			
		Konkani	Hindi	Marathi	Nepali
90	थू	377	0	0	0
91	स्तेँ	373	10	1	0
92	स्नां	365	0	0	1
93	ळळा	364	0	0	0
94	श्री	364	0	2	39
95	ड्ड्या	314	0	22	0
96	स्थें	313	0	3	1
97	स्टॅ	311	0	173	0
98	ष्ट्रां	305	0	1310	0
99	ज्ञां	288	0	608	0
100	न्नां	284	0	3	0
101	ल्सां	284	0	0	0
102	प्टें	282	16	1155	0
103	त्शें	277	0	0	0
104	ध्दा	266	0	916	42
105	ळळां	265	0	0	0
106	र्भा	262	0	24	0
107	र्था	258	0	510	2
108	ब्रॅ	256	0	40	0
109	त्यार्	249	0	263	1
110	फ्रॅ	245	0	18	5
111	ख्ये	237	0	1304	9
112	ल्मां	233	30	0	4
113	ग्दा	221	0	23	533
114	ड्डो	219	6	0	2
115	स्त्यां	210	0	334	0
116	कीं	208	1	934	0
117	शठा	204	0	0	6
118	त्रॉ	202	0	0	0

Sr. No.	Syllables(<i>ligature-vowels-signs-diacritic</i>)	Frequency			
		Konkani	Hindi	Marathi	Nepali
119	न्हू	200	1	0	12
120	र्डा	199	0	14	0
121	म्हों	198	2	0	0
122	ळ	197	0	0	1
123	शिट्टे	192	0	0	4
124	फ़	191	0	41	0
125	स्पर्	191	0	27	0
126	स्तीं	189	0	8	4
127	त्म्या	185	0	638	2
128	द्रे	184	0	0	4
129	के	180	0	0	0
130	चे	180	12	0	0
131	व्हं	179	0	0	0
132	दीं	177	0	18	4
133	त्शी	175	3	1	0
134	ळीं	175	0	0	0

Table A.4: Example of syllables (*ligature-vowels-signs-diacritic*) and their frequency in Konkani, Hindi, Marathi and Nepali

A.5 Some example of *ligatures* whose occurrence is zero in one of the languages

<i>Ligatures</i>	Frequency				<i>Ligatures</i>	Frequency			
	Konkani	Hindi	Marathi	Nepali		Konkani	Hindi	Marathi	Nepali
ळ्य	14786	0	6238	1	प्य	42	0	851	1
ळ	5132	0	6	3	स्त	42	0	0	2

Ligatures	Frequency				Ligatures	Frequency			
	Konkani	Hindi	Marathi	Nepali		Konkani	Hindi	Marathi	Nepali
ळळ्य	868	0	0	0	र्झ	39	0	5	0
श्श	254	0	851	9	त्स्फ	35	0	18	0
ण्ट्य	187	0	0	0	ण्ड्र्य	35	0	0	0
न्न	177	0	0	11	य्क	33	0	1	11
ट्ट्य	164	0	206	45	ज्य	30	0	7	1
क्क्य	163	0	1067	0	ल्य	27	0	8	25
इ	138	0	31	64	श्द्र	26	0	0	0
श्ण	130	0	0	6	ल्ढ	26	0	0	9
द्र्य	121	0	69	0	ण्ड्	26	0	0	1
ज्य	115	0	9	19	ङ्ग	24	0	0	17885
ख्ख्य	105	0	8	0	ण्ड्स	23	6	0	33
र्झ	103	0	40	39	श्द	23	0	40	0
क्झ	85	0	108	4	क्य	22	0	17	38
र्ध	81	0	5	3	श्क्र	21	0	0	2
द्घ	74	0	221	6	र्द	20	0	4	2
य्ल	73	0	0	6	व्ह्य	19	0	24	0
त्झ	66	0	69	0	ल्झ	18	0	7	111
ट्य	64	0	1	1	म्हज्य	17	0	0	0
व्य	61	0	0	0	य्ट	17	0	0	2
श्ट्य	55	0	0	0	व्ण	17	0	0	0
ग्य	53	0	19	5	र्श	17	0	12	1
श्र्य	51	0	96	0	ण्ड्य	17	0	0	57
ड्ट	47	3	0	0	ज्झ	16	0	341	1
ल्म्य	47	0	0	4	न्व्ह	16	0	133	0
व्क	46	0	2	1	म्ब्य	16	0	8	21
ल्व्ह	46	0	25	0	ग्झ	16	0	1	0
ट्ट्	43	0	0	2	स्ए	16	1	0	6

Table A.5: Some example of *ligatures* whose occurrence is zero in one of the languages

Appendix B

Example of errors introduced due to use of Hindi OCR on Konkani documents

B.1 Top 90 syllables contributed to errors along with its frequency, error % and top 6 OCR recognized substitutions.

#	Actual Syllable	Description of syllable in column 2	Error freq.	Error %	Top 6 OCR substitutions with frequency					
					1	2	3	4	5	6
1	ଌ	Consonant ଌ ([l̪ə])	218565	1.24%	ଌ (39922)	ଌ (21295)	ଌ (16601)	ଌ (13289)	ଌ (9178)	ଌ (7286)
2	ଌ̣	Anusvara (nasal)	49247	0.2794%	<SPC>(47284)	ଌ̣ (1269)	ଌ̣ (487)	ଌ̣ (162)	ଌ̣ (15)	ଌ̣ (6)

#	Actual Syllable	Description of syllable in column 2	Error freq.	Error %	Top 6 OCR substitutions with frequency					
					1	2	3	4	5	6
3	ँ	vowel sound [ɛ]	48242	0.2737%	ेँ (33358)	ैँ (1625)	ँ(524)	ं (512)	ेँँ (182)	ँँ (98)
4	'	[U+2019] (right single quotation mark)	20468	0.1161%	<SPC> (12025)	" [U+201D] (2346)	! (2185)	* (1406)	ः (761)	!(645)
5	'	[U+2018] (left single quotation mark)	19496	0.1106%	" (13113)	'[U+0027] (4580)	<SPC> (472)	""0 (472)	" (262)	* (175)
6	र	Consonant र ([rə])	10800	0.0613%	न (6832)	च (2756)	<SPC>(350)	य (171)	व (137)	त (112)
7	.	dot	10522	0.0597%	, (10485)	– (19)	: (7)	<SPC> (3)	ि (2)	ी (2)
8	े	Vowel sign े	9562	0.0542%	ै (7599)	<SPC>(1091)	ि (447)	ु (46)	ा (15)	ुं (14)
9	ळ	Consonant ळ ([ɭ]) with vowel sign ा ([a])	8224	0.0467%	ब्ण (1167)	व्ग (702)	व्ण (666)	ब्श (657)	ब्ह (456)	ढ्य (249)
10	व	Consonant व ([və])	6186	0.0351%	ब (5316)	ल (316)	क (202)	त (86)	त (86)	<SPC>(84)
11	''	[U+2019] [U+2019] (two right single quotation mark)	4577	0.026%	" [U+201d] (585)	"" (374)	' (272)	(167)	"" (98)	" (90)
12	त्	Consonant त ([tə]) with <i>halant</i> sign ्	3578	0.0203%	<SPC>(3568)	ट (2)	न (2)	ञ (2)	स (2)	2(2)
13	ह्र	Consonant ह ([ɦ]) with <i>halant</i> sign ् and consonant य ([jə])	2913	0.0165%	द (1491)	दव (903)	झ (387)	दाव (87)	छ (15)	दम (12)
14	र्	Consonant र [rə] with <i>halant</i> sign ् (mostly used with ZWJ)	2657	0.0151%	र (1536)	र् (1056)	र् (56)	र् (3)	र् (2)	र (2)
15	न	Consonant न ([nə])	2438	0.0138%	र (1100)	ञ (717)	ज (265)	त (166)	व (42)	म (24)

#	Actual Syllable	Description of syllable in column 2	Error freq.	Error %	Top 6 OCR substitutions with frequency					
					1	2	3	4	5	6
16	ळ	Consonant ळ ([ɭ]) with <i>halant</i> sign ् and consonant ळ ([ɭ])	2314	0.0131%	लूक (249)	ठक (156)	नूक (141)	ल (135)	रूक (108)	नक (105)
17	य	Consonant य ([jə])	1977	0.0112%	<SPC>(1159)	व (275)	म (97)	प (90)	च (77)	स (57)
18	ह	Consonant ह ([ɦə])	1468	0.0083%	<SPC>(913)	द (441)	ट (73)	च (8)	टद (8)	ड् (4)
19	र्	Consonant र ([rə]) with <i>halant</i> sign ्	1466	0.0083%	<SPC>(1328)	व (54)	च (20)	ग (8)	ल (6)	क (6)
20	ल	Consonant ल ([lə])	1462	0.0083%	ह (334)	<SPC>(274)	न (218)	द (139)	र (123)	व (66)
21	अँ	Vowel अ with vowel sign ँ ([ɛ̃])	1238	0.007%	ऑ (936)	औ (60)	ऑँ (52)	ऑं (48)	ओं (34)	ओ (28)
22	ू	Vowel sign ू ([uː])	1160	0.0066%	ृ (441)	ु (351)	्ड (92)	े (86)		
23	ट	Consonant ट ([ʈə])	1094	0.0062%	द (386)	ड (285)	ठ (263)	र (35)	ह (26)	द्व्ठ (9)
24	ा	Vowel sign ा ([aː])	964	0.0055%	ॉ (555)	<SPC>(326)	ो (28)	े (17)		
25	प	Consonant sign प ([pə])	941	0.0053%	ष (697)	य (58)	छ (26)	घ (25)	ज (22)	ए (19)
26	ळो	Consonant ळ ([ɭ]) with vowel sign ो ([o])	930	0.0053%	बग्े (152)	व्शे (48)	<SPC> (46)	ब्ण्े (44)	व्ण्े (36)	त्ण्े (32)
27	ळ्	Consonant ळ [ɭ] with <i>halant</i> sign ्	858	0.21%	नू (368)	लू (112)	रू (68)	छू (62)	व (50)	न (30)
28	अ	Vowel अ ([ə])	778	0.0044%	आ (509)	थ (58)	थर (38)	थआ (36)	आर (24)	ज (20)
29	ड	Consonant ड ([ɖə])	766	0.0043%	द (352)	ह (123)	<SPC>(107)	र (31)	झ (19)	उ (14)
30	त	Consonant त ([tə])	736	0.0042%	ज (389)	<SPC>(211)	द (28)	क (27)	ट (24)	न (11)
31	"	[U+201C] (left double quotation mark)	735	0.0042%	> * (28)	'(10)	<SPC> (8)	"* (2)	>ब (2)	> (2)

#	Actual Syllable	Description of syllable in column 2	Error freq.	Error %	Top 6 OCR substitutions with frequency					
					1	2	3	4	5	6
32	कां	Consonant क ([kə]) with vowel sign ा ([a:]) and with Anusvara ँ	724	0.0041%	बग्' (284)	व्ण' (104)	ब्य' (40)	व्य' (36)	दण' (24)	<SPC>(24)
33	ठ	Consonant letter ठ ([tʰə])	710	0.004%	ड (284)	ट (217)	द (57)	व (30)	त (17)	<SPC>(14)
34	स	Consonant स ([sə])	699	0.004%	ल (381)	र (126)	ख (81)	म (38)	<SPC>(30)	ज (19)
35	क	Consonant क ([kʰə])	688	0.0039%	व (241)	त (200)	ब (79)	न (36)	ल (34)	<SPC>(28)
36	र	Consonant र ([rə])	661	0.0038%	र (555)	रु (88)	रू (6)	रं (4)	रु (2)	रू (2)
37	ळे	Consonant ल ([lə]) with vowel sign े ([e])	600	0.0034%	<SPC>(108)	ढ् (94)	ल् (84)	क् (68)	न् (42)	ढ्ठ (33)
38	ज	Consonant ज ([dʒə])	555	0.0031%	ञ (412)	<SPC> (73)	ल (30)	न (15)		
39	ँ	Sign <i>candrabindu</i> (anunasika)	545	0.0031%	ैं (364)	<SPC> (87)	ै (50)	े (22)	ं (4)	र् (4)
40	ह्	Consonant ह ([ɦə]) with <i>halant</i> sign ्	516	0.0029%	द (504)	टदा (3)	दा (2)	दृ (2)	दृ (2)	छा (2)
41	श	Consonant श ([ʃə])	472	0.0027%	र (305)	ल (85)	<SPC>(11)	क (9)	ण (9)	ह (7)
42	य	Consonant य ([jə])	456	0.0026%	ग्र (168)	य (60)	ग्न (36)	य (35)	य् (22)	श (16)
43	त्र	Consonant त ([tə]) with <i>halant</i> sign ् and consonant र ([rə])	447	0.0025%	न (198)	<SPC>(132)	ज (69)	रु (24)	नज (12)	च (9)
44	झ	Consonant झ ([dʒʰə])	390	0.0022%	ल्ल (27)	ज (26)	म (23)	ज्ज (21)	शन् (18)	ह (18)
45	ं	Sign Nukta	366	0.0021%	<SPC>(297)	्र (54)	्र (7)	्रु (2)	ड (2)	ृ (1)
46	ण	Consonant ण ([ɳə])	357	0.002%	ग (116)	प (95)	्र (70)	रु (32)	र (17)	<SPC> (9)
47	?	Question mark	302	0.0017%	श (192)	! (22)	<SPC>(17)	१ (9)	) (7)	ः (4)
48	म	Consonant म ([mə])	286	0.0016%	ग (144)	भ (56)	स (14)	य (13)	<SPC> (6)	प (6)

#	Actual Syllable	Description of syllable in column 2	Error freq.	Error %	Top 6 OCR substitutions with frequency					
					1	2	3	4	5	6
49	ळळ	two Consonant ळ ([ɭə])	285	0.0016%	छक (76)	कक (14)	छक् (12)	छढ (12)	लक (10)	छछ (10)
50	ळीं	Consonant ळ ([ɭə]) with vowel sign ी ([i:]) and Anusvara ँ	263	0.0015%	रुढ् (16)	<SPC> (15)	ई (12)	रु् (9)	र्न (5)	रुब् (5)
51	ब	Consonant ब	263	0.0015%	ल (83)	व (60)	ज (54)	त (12)	द (10)	<SPC> (8)
52	ऐ	Vowel letter <i>candra</i> [ɛ] ऐ	254	0.0014%	ऐं (158)	ऐ (18)	एँ (16)	ऐँ (16)	एं (16)	एऐं (12)
53	द	Consonant द ([d̪ə])	247	0.0014%	च (56)	ह (47)	ठ (32)	ग (22)	ग (22)	व (15)
54	ध	Consonant ध ([d̪ʰə])	237	0.0013%	थ (207)	श (16)	<SPC> (6)	व (3)	द (2)	ष (2)
55	ळू	Consonant ळ ([ɭə]) with vowel sign ू ([u:])	208	0.0012%	दुद (33)	<SPC> (22)	द (20)	ल (20)	दुध (18)	ूँ (8)
56	च	Consonant च ([tʃə])	177	0.001%	य (51)	<SPC> (29)	द (28)	ब (9)	र (8)	ह (7)
57	लब	Two consonant ल ([lə]) and ब ([bə])	162	0.0009%	न (82)	नन (76)	न (2)	क (2)		
58	रा	Consonant र with vowel sign ा ([a])	158	0.0009%	ग (144)	<SPC> (6)	श (2)	स (2)	ग (2)	पु (2)
59	अु	Vowel अ ([ə]) with vowel sign ु ([u])	150	0.0009%	आ (132)	आ (4)	ओओ (4)	श (2)	आओ (2)	<SPC> (2)
60	द्य	Consonant द ([d̪ə]) with <i>halant</i> ण् and consonant य [jə]	147	0.0008%	च (135)	<SPC> (6)	चध (3)	ध (3)		
61	डु	consonant ड ([d̪ə]) with <i>halant</i> ण् and consonant उ ([u])	141	0.0008%	इ (132)	इ (9)				
62	ड	Consonant ड ([d̪ə])	132	0.0007%	ड (81)	ह (10)	ड (10)	झ (4)	झ (4)	ज (4)
63	ढ	Consonant ढ ([d̪ʰə])	129	0.0007%	ब (58)	ब (41)	द (8)	क (6)	ड (3)	दु (3)
64	ग	Consonant ग ([gə])	122	0.0007%	म (51)	ब (9)	ण (9)	प (7)	श (7)	ु (6)

#	Actual Syllable	Description of syllable in column 2	Error freq.	Error %	Top 6 OCR substitutions with frequency					
					1	2	3	4	5	6
65	थ	Consonant थ ([tʰə])	120	0.0007%	श (95)	<SPC> (10)	ध (5)	भ (4)	् (3)	घ (1)
66	घ	Consonant घ (gʱə])	116	0.0007%	च (28)	प (24)	श (16)	ज (14)	ष (12)	न (5)
67	ळ्	Consonant ळ ([ɭ]) with <i>halant</i> ् and consonant ळ ([ɭ]) with <i>halant</i> ्	108	0.0006%	नूवू (12)	नूनू (12)	ठक (8)	लूकू (8)	ठव (8)	लन (8)
68	ट्ट	Consonant ट ([tə]) with <i>halant</i> ् and consonant ट ([tə])	108	0.0006%	इ (90)	इल (3)	इड (3)	डल (3)	हड (3)	ह (6)
69	ओ	Vowel ओ ([o])	85	0.0005%	ऑ (83)	आ (2)				
70	ड्	Consonant ड ([də]) with <i>halant</i> ्	84	0.0005%	इ (76)	<SPC> (6)	य (2)			
71	झर्	Consonant झ ([dzʰə]) and consonant character रं ([rə]) with <i>halant</i> ्	69	0.0004%	ई (30)	य (21)	ड (9)	ल (6)	श (3)	
72	ळि	Consonant ळ ([ɭ]) with vowel sign ि ([i])	66	0.0004%	ल् (28)	ब् (14)	क् (8)	ढ (4)	ल (2)	ढ् (2)
73	वृ	consonant द ([dʱə]) with vowel sign Vocalic R ृ	66	0.0004%	ह (46)	ह् (10)	ह्र (4)	इ (2)	इह (2)	हु (2)
74	टु	consonant ट ([tə]) with vowel sign ु ([u])	66	0.0004%	दृठ (18)	ठ (10)	ठू (8)	डू (8)	दू (6)	ड (4)
75	या	consonant य ([jə]) with vowel sign ा ([a])	54	0.0003%	<SPC> (40)	आ (8)	ाव (4)	व (2)		
76	लँ	Consonant ल ([lə]) and vowel sign ँ [ɛ]	42	0.0002%	वें (12)	रै (6)	रैं (6)	ुँ (4)	रैँ (4)	रुँ (3)
77	!	!	38	0.0002%	। (8)	<SPC> (7)	/ (5)	((6)	1 (6)	श (6)
78	रु	Consonant र ([rə]) and vowel sign ु ([u])	34	0.0002%	<SPC> (12)	ं (8)	्र (6)	ल (4)	स (2)	स् (2)

#	Actual Syllable	Description of syllable in column 2	Error freq.	Error %	Top 6 OCR substitutions with frequency					
					1	2	3	4	5	6
79	ड्य	Consonant ड with <i>halant</i> ् and consonant य [jə]	33	0.0002%	द (21)	ग (9)	झ (3)			
80	यो	Consonant य ([jə]) with vowel sign ो ([o])	32	0.0002%	ा (12)	ोँ (12)	व (4)	<SPC> (4)	ाा (2)	
81	ज	Consonant ज ([jə])	32	0.0002%	ज (18)	<SPC> (6)	आ (6)	ं (2)		
82	–	Hyphen	30	0.0002%	<SPC> (15)	. (4)	र (3)	न (2)	ए (1)	श (1)
83	भ	Consonant भ ([b ^h ə])	29	0.0002%	म (18)	ध (4)	ेम (2)	न (1)	त (1)	क (1)
84	3	Number 3	27	0.0002%	३ (20)	३, (2)	\$ (2)	14 (2)		
85	ळय	Consonant ळ ([lə]) and consonant य ([jə])	26	0.0001%	छणयन् (5)	छयन् (5)	तय् (4)	ढय् (4)	छय् (4)	छगन् (4)
86	न	Consonant न ([nə])	26	0.0001%	जन् (6)	जञ् (6)	नन् (6)	न (4)	न् (2)	जं (2)
87	आ	Vowel आ ([a])	25	0.0001%	अ (13)	क्षा (8)	शअ (2)	ऑ (1)	<SPC> (1)	
88	उ	Vowel उ ([u])	25	0.0001%	ऊ (6)	ज (5)	ड (4)	ठ (3)	र (2)	ए (1)
89	7	Number 7	25	0.0001%	9 (8)	प् (6)	1 (4)	एप् (3)		
90	ष	Consonant ष ([ʃə])	20	0.0001%	ट (3)	<SPC> (3)	घ (3)	छ (2)	ल (2)	र (2)

Table B.1: Top 90 syllables contributed to errors along with its frequency, error % and top 6 OCR recognized substitutions.

B.2 Top 50 Konkani words misrecognized by Tesseract OCR

Sr. No.	Actual word	Freq.	%	Sr. No.	Actual word	Freq.	%
1	ह्या (this)	8907	45.0 %	26	मॅची (matches)	1294	0%
2	हें (this)	6816	33.0%	27	यत्न (attempt)	1148	7.0%
3	न्हय (no/river)	4816	12.0%	28	पळोवंक (see)	1136	0%
4	वेळार(at the time)	4618	0%	29	खेळां (games)	1093	0%
5	हे (this)	3926	41.0%	30	टॅस्ट (test)	1063	0%
6	वर्सा (years)	3625	0%	31	सगळ्यांत (among)	1041	0%
7	अशें (like this)	3237	51.0%	32	मेळयलें (meet)	1030	0%
8	मॅच (match)	2532	0%	33	पळय (see)	977	0%
9	मदीं (middle)	2515	3.0%	34	दोळे (eyes)	973	0%
10	म्हळ्यार (means)	2395	0%	35	काळांत (era)	951	0%
11	सर्तींत (in bet)	2387	0%	36	सकाळीं (morning)	911	0%
12	पळोवन (seen)	2318	0%	37	जाल्ल्या (happen)	872	3.0%
13	वेळ (time)	2287	0%	38	मेळोवन (met)	868	0%
14	धांवड्यो (runner)	2053	30.0%	39	विशीं (topic)	841	23.0%
15	मेळटा (meets)	2003	0%	40	तुवें (you)	827	13.0%
16	म्हळें (said)	1987	0%	41	मळार (field)	812	0%
17	सगळे (all)	1977	0%	42	यत्न(attempt)	806	0%
18	सगळें (all)	1887	0%	43	न्ही (no)	776	0%
19	मॅचींत (in game)	1766	0%	44	मेळूंक (got)	772	0%
20	सगळ्या (all)	1705	0%	45	वळख (familiar)	736	0%
21	खेळ (game)	1669	0%	46	सगळीं (all)	733	0%
22	दुसऱ्या (other)	1613	0%	47	मेळना (not got)	719	0%
23	आदीं (before)	1388	7.0%	48	पळयलें (seen)	680	0%
24	फाटीं (back)	1367	0%	49	मूळ (root)	678	0%
25	आतां (now)	1338	34.0%	50	मेळटात (meet)	678	0%

Table B.2: Top 50 frequently misrecognized actual Words, their frequency and accuracy in percentage.

Appendix C

Example of word and syllables correctly and incorrectly recognized by *Konkani_Multi_Font* OCR

C.1 Most frequent words recognized correctly by *Konkani_Multi_Font* OCR for all ten fonts (i.e. *Amiko, Lohit Devanagari, Biryani, Nakula, Cambay, Gargi, Asar and Baloo and Aparajita*)

Sr. No.	Actual word	Gloss	freq	Sr. No.	Actual word	Gloss	freq
1	हे	This	999	26	तरी	Atleast	294
2	तांणी	They	734	27	हांचे	His	294
3	एक	One	693	28	चड	More	287
4	पूण	But	693	29	आसात	Having	273
5	कडेन	Near by	686	30	तीन	three	273
6	जाल्यार	Condition	599	31	नायक	Hero	273
7	ही	she	588	32	मुखार	Forward	273
8	हो	he	579	33	उमेदवार	Candidate	268
9	आड	Bring	560	34	तें	This	266
10	उपरांत	Afterwards	546	35	वा	Wow	266

Sr. No.	Actual word	Gloss	freq	Sr. No.	Actual word	Gloss	freq
11	करून	After doing	454	36	क्रिकेट	Cricket	259
12	वेळार	On time	399	37	असो	Like this	252
13	अशी	This way	398	38	जातली	It will be done	252
14	आमी	We	385	39	निर्णय	Decision	252
15	तशेंच	Like this	384	40	लागून	Because of	252
16	म्हायती	Information	364	41	उमेदवारा	Candidates	251
17	आपल्या	Ours	357	42	आतां	Now	245
18	गोंय	Goa	343	43	करतले	Will do	245
19	जाता	Happening	342	44	ती	They	245
20	म्हणल्या	Means what	336	45	काल	Yesterday	238
21	हेर	Others	335	46	घेवन	Taken	238
22	ताका	For him	322	47	प्रशिक्ष	Exam	238
23	जैत	Win	315	48	सुरू	To start	238
24	वेंचणूक	Election	306	49	तांचे	His	231
25	करपाक	To do	301	50	मात	Nothing	231

Table C.1: Most frequent words which are always recognised correctly by *Konkani_Multi_Font* OCR on the *Konkani_test_dataset* for all ten fonts

C.2 Substitution errors exhibited by *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *Category-I* (i.e. *Amiko, Lohit Devanagari, Biryani, Nakula, Cambay, Gargi and Asar*) fonts

This Table has five columns. The first column contains the actual character (AC), second column has the total frequency of the characters being not recognized correctly for the fonts within their respective categories. Third and fourth column has top 2 recognized character by OCR indicated as (RC), along with its occurrence (freq) which exhibited errors. The fifth column has the average

error percentage for the fonts within each category; it is indicated as (%), which is also shown in Table 8.7 (refer Chapter 8).

AC	freq	RC (freq)		%	AC	freq	RC (freq)		%
Devanagari Signs and other characters					Devanagari Dependent vowels				
'	664	' (612)	<SPC> (28)	0.102	ँ	444	ँ (168)	ै (145)	0.077
'	548	" (274)	<SPC> (194)		ॉ	275	ॉ (100)	ा (75)	
ं	118	. (86)	<SPC> (22)		ां	126	ो (124)	ा (2)	
ँ	53	ं (26)	ँ (18)		े	72	ो (46)	ि (9)	
)	33	<SPC> (33)	-		ु	52	ू (42)	् (1)	
''	16	" (6)	' (6)		ॉ	48	ॉ (24)	ौ (16)	
,	8	<SPC> (6)	. (2)		ू	33	ृ (14)	् (8)	
:	8	ः (8)	श (1)		ो	32	ी (16)	े (5)	
?	4	श (1)	1 (1)		ा	15	ो (9)	ी (6)	
०	3	० (2)	् (1)		ृ	11	ू (11)	-	
Devanagari consonants					ि	6	े (5)	् (1)	
व	152	ब (121)	ठ (6)	0.066	ी	2	ा (2)	-	
न	139	र (125)	ज (5)		Independent vowels				
ल	112	ु (15)	त् (14)		ँ	598	ं (396)	एँ (57)	0.044
य	105	य (य+ ँ) (42)	य (26)		ऑ	47	अ (24)	आ (21)	
ट	93	ड (31)	ठ (30)		उ	7	ऊ (1)	ज (6)	
र	73	न (19)	स (15)		आ	4	अ (2)	अन (2)	
त	62	ब (25)	ज (11)		इ	1	ड (1)		
द	39	ट (31)	ढ (4)		syllables/conjunct consonants				
ड	23	ट (18)	इ (3)		ठ	42	ढ (18)	ढट (12)	0.017
र	23	र (23)	-		द्व	30	ह (27)	ह (3)	
श	21	क (4)	(2)		डु	26	डु (9)	हु (9)	
प	20	ष (12)	म (4)		द्व	24	ड (6)	द्व (3)	
ज	16	ज (9)	ल्या (4)		द्व	12	इ (10)	द (2)	
य	15	घ (1)	थ (1)		डे	12	ट (6)	टु (6)	
ग	13	य (7)	प (1)		त्र	12	न (12)	-	
ळ	10	ढ (4)	व्य (3)		फ़ा	12	छार (12)	-	

AC	freq	RC (freq)		%	AC	freq	RC (freq)		%
ख	8	र (2)	ख (2)		फ़	12	छुं (12)	-	
ध	8	घ (4)	थ (4)		क्क	9	छ (9)	-	
क	7	व (3)	न (1)		डें	9	ट (6)	रु (3)	
म	7	ग (5)	श (1)		फ़	9	छ (9)	-	
घ	4	ग (1)	ठ (1)		रा	6	ण (6)	ग (2)	
च	4	व (3)	ब (1)		क्य	6	वब (6)	-	
ण	4	प (3)	ग (1)		रु	6	खु (2)	सु (2)	
फ	4	छ (3)	क (1)		र्व	5	वजीत (5)	-	
स	3	रु (2)	झ (1)		ख	4	ख (4)	-	
ब	2	न (2)	-		राच	4	यय (4)	-	
ड	1	ड (1)	-		ठु	3	ढ (3)	-	
छ	1	झ (1)	-		टँ	2	ड (2)	-	
भ	1	श (1)	-	रड	2	ख (2)	-		
AC: Actual Character					रा	2	र (2)	-	
RC: Recognized Character					त्शे	4	ळी (4)	-	
freq: Frequency, <SPC>: space character					स	9	तस (9)	-	

Table C.2: Substitution errors exhibited by *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *Category-I* fonts (i.e. *Amiko*, *Lohit Devanagari*, *Biryani*, *Nakula*, *Cambay*, *Gargi* and *Asar* fonts)

C.3 Substitution errors exhibited by *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *Category-II* (*Baloo* and *Apara-jita*) fonts

AC	freq	RC (freq)		%	AC	freq	RC (freq)		%
Devanagari consonants					Devanagari signs and other characters				
ह	1791	ड (1348)	ढ (266)		'	163	" (87)	" (36)	0.063

AC	freq	RC (freq)		%	AC	freq	RC (freq)		%		
द	338	ठ (243)	ठ (32)	0.681	ँ <i>Anusvara</i>	92	<SPC> (66)	् (7)			
न	260	र (120)	त (77)		ँ	66	ं (55)	ँ (7)			
भ	181	म (147)	म (24)		?	12	श् (4)	श (2)			
व	94	ब (68)	ब (8)		. dot	7	(4)	(3)			
र	90	य (61)	स (7)		ं <i>Nukta</i> ¹	6	् (4)	् (2)			
य	55	य (य+ ं ¹)(30)	यम्र (4)		''	6	" (4)	" (2)			
च	42	व (41)	य (1)		,	3	. (2)	(1)			
ल	41	त् (14)	त (9)		Independent vowels						
ट	37	द (13)	व (9)		ँ	217	ं (122)	ए (21)	0.062		
ड	30	ट (19)	इ (4)		औ	137	आं (112)	अ (14)			
ख	29	रब (14)	रव (10)		इ	4	ड (2)	ड् (2)			
य	18	प (12)	थ (2)		Syllables/Conjunct consonants						
ठ	17	ढ (7)	ढट (4)		रा	849	य (798)	य (22)	0.031		
त	10	न (3)	ठ (2)		रौ	12	यँ (8)	सँ (4)			
ग	9	य (3)	प (2)		डू	10	इ (8)	द (2)			
ळ	9	न (3)	व्य (3)		हौ	10	डे (8)	डँ (2)			
र	8	र (6)	ख (2)		शा	9	वे (6)	? (3)			
घ	6	ध (3)	य (3)		डै	6	ट (6)	-			
थ	3	यश (2)	श (1)		ट्र	5	द्र (3)	ह (2)			
स	3	झ (1)	य (1)		डे	4	ट (4)	-	0.012		
प	2	ष (9)	फ (1)		Numbers						
ढ	1	ड (1)	-		3	16	३ (16)	-			
Devanagari Dependent vowels					0	8	० (8)	-			
ँ	213	ा (76)	ां (76)	5	8	५ (8)	-				
ँ	177	ं (74)	ँ (28)	3	6	३ (6)	-				
ो	64	े (50)	ये (8)	5	6	५ (6)	-				
ु	62	ू (39)	् (6)	30	3	३० (3)	-				
ा	49	य (46)	(2)	35	3	३५ (3)	-				
े	49	ो (38)	ि (9)	1	2	(2)	-				
ू	30	ृ (12)	् (8)	9	2	९ (2)	-				

¹ <https://www.unicode.org/charts/PDF/U0900.pdf>

AC	freq	RC (freq)		%	AC	freq	RC (freq)		%
ँ	16	ँ (8)	ी (2)		9	2	१ (2)	-	
ै	9	े (7)	ेग (2)		31	2	उ (2)	-	
ृ	5	ू (1)	ड (2)		55	2	ऽ (2)	-	
ौ	5	ो (5)	-		3	1	उ (1)	-	
AC: Actual Character RC: Recognized Character freq: Frequency, <SPC> space character									

Table C.3: Substitution errors exhibited by *Konkani_Multi_Font* OCR on *Konkani_test_dataset* for *Category-II* fonts (i.e. *Aparajita* and *Baloo* fonts)

C.4 Words exhibit zero accuracy for all ten fonts (i.e. *Amiko*, *Lohit Devanagari*, *Biryani*, *Nakula*, *Cambay*, *Gargi*, *Asar*, *Baloo* and *Aparajita* fonts)

The table has six columns. The first column represent the Sr.No indicated as #, second column contains the actual word, third column has its Roman IAST transliteration, which is done using Aksharamukha Transliteration tool², fourth column has gloss of same, fifth column has frequency indicated as freq and sixth column consist of the corresponding output of *Konkani_Multi_Font* OCR.

Sr. No.	Actual word	Roman (IAST)	Gloss	frequency	Recognised word
1	वर्सा	varsām	Years	270	वर्सा
2	अर्दात	ardāṃta	In the middle	40	अर्दात
3	सर्तीतल	sartīṃtala	In the competition	30	सर्तीतल
4	पार्से	pārse	Parse	20	पार्स
5	कोर्टात	koṛṭāṃta	In the courtroom	10	कोर्टात
6	एँड	æḍa	And	100	एँड

²<https://aksharamukha.appspot.com/converter>

Sr. No.	Actual word	Roman (IAST)	Gloss	frequency	Recognised word
7	एडलेंड	æḍalæḍa	Adelaide	40	एडलेंड
8	ऐसेंब्ली	æseṃblī	Assembly	30	ऐसेंब्ली
9	ऐण्डी	æṇḍī	Andy	30	ऐण्डी
10	एडलेंडांत	æḍalæḍāṃta	In Adelaide	20	एडलेंडांत
11	ऐलेक्झें	ælekjhem	Alexan	10	ऐलेक्झें
12	ऐपल	æpala	Apple	10	ऐपल
13	ऐन्थनी	ænthanī	Anthony	10	ऐन्थनी
14	ऐन्टिक्क	æṇṭikka	Antique	10	ऐन्टिक्क
15	ऐण्डर्सन	æṇḍarsana	Anderson	10	ऐण्डर्सन
16	एडलेडांत	æḍaleḍāṃta	In Adelaide	10	एडलेडांत
17	एडमिशन	æḍamiśana	Admission	10	एडमिशन
18	ऐक्केरिय	ækkeriya	Ackery	10	ऐक्केरिय
19	अड	æḍa	Adv	170	अड
20	अडव्होक	æḍavhoka	Advoc	40	अडव्होक
21	अॅलोपॅथि	ælopæthi	Allopathy	10	अलोपथ
22	एकगड्डो	ekagaṭṭho	A bundle	30	एकगड्डो
23	रट्	raṭ	Rat	20	ट
24	मेळळ्या	mellyā	has been found	10	मेळळ्या
25	उक्ते	uktem	open	10	उक्ते
26	आयलो	āyalo	To come	10	आयलो
27	आयची	āyacī	Today's	10	आयची
28	आयएएस	āyaeesa	IAS	10	आयएएस
29	आसता'	āsata	Existing	10	आसता
30	इत्सा'	itsā	will	10	इत्सा'
31	दर्शन'	darśana	Philosophy	20	दर्शन'
32	क्लीन'	klīna	Clean	10	छीन
33	'भक्त	bhakta	devotee	30	"भक्त
34	'पेड	peḍa	Ped	50	"पेड
35	'झकास'	jhakāsa	Jhakas	30	"झकास!
36	'गोमंत	gomaṃta	Gomantak	30	"गोमंत

Sr. No.	Actual word	Roman (IAST)	Gloss	frequency	Recognised word
37	कोठयेर'	koṭhayera	Kothyer	10	कोठयेर'
38	'आरे	'āre	'Hey	10	आरे

Table C.4: Words exhibit zero accuracy for all ten font (i.e. *Amiko*, *Lohit Devanagari*, *Biryani*, *Nakula*, *Cambay*, *Gargi*, *Asar* and *Baloo* and *Aparajita* fonts)

References

- Banerjee, A., Jain, A., Mhaskar, S., Dattatray Deoghare, S., Sehgal, A., Bhattacharya, P., Aug. 2021. Neural Machine Translation in Low-Resource Setting: a Case Study in English-Marathi Pair. In: Proceedings of Machine Translation Summit XVIII: Research Track. Association for Machine Translation in the Americas, Virtual, pp. 35–47.
URL <https://aclanthology.org/2021.mtsummit-research.4>
- Bansal, V., Sinha, M., 2001. A complete OCR for printed Hindi text in Devanagari script. In: Proceedings of Sixth International Conference on Document Analysis and Recognition. IEEE, Seattle, WA, USA, pp. 800–804.
URL <https://doi.org/10.1109/ICDAR.2001.953898>
- Bhave, B., Borkar, S., Karmali, N., Halankar, T., Ghanekar, D., Borges, S., Thali, M., Nayak, B., Tadakodkar, P., Pandit, A., Kalangutkar, V., 2015. Konkani Shuddhalekhanache Nem 5th Edition. Secretary, Goa Konkani Academy, 243, Patto Colony, Panaji , Goa - 403 001.
- cfilt.iitb, 2023. Word Frequency for Hindi. Indian Institute of Technology, Bombay, (accessed July 14, 2023).
URL <http://www.cfilt.iitb.ac.in/Downloads.html>
- Consortium, U., 2022. The Unicode Standard Version 15.0 – Core Specification. Mountain View, CA.
URL <https://www.unicode.org/versions/Unicode15.0.0/UnicodeStandard-15.0.pdf>
- Dalvi, G., 2019. Terminology of Devanagari Typefaces. (accessed January 20, 2019).
URL <http://dsource.in/tool/devft/en/terminology.php>
- Das, D., Philip, J., Mathew, M., Jawahar, C., 2019. A cost efficient approach to correct OCR errors in large document collections. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, IEEE, Sydney, NSW, Australia, pp. 655–662.
URL <https://doi.org/10.1109/ICDAR.2019.00110>

- Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., Jun. 2019. 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, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186.
URL <https://aclanthology.org/N19-1423>
- Ek-Type, D., 2019. Baloo fonts. (accessed 24 January , 2019).
URL <https://fonts.google.com/specimen/Baloo>
- font Biryani, G., 2019. Biryani fonts. (accessed 24 January , 2019).
URL <https://fonts.google.com/specimen/Biryani>
- Gaurav, Dec 2019a. inltk nepali news dataset. <https://www.kaggle.com/datasets/disisbig/nepali-news-dataset>.
URL <https://www.kaggle.com/datasets/disisbig/nepali-news-dataset>
- Gaurav, Dec 2019b. Nepali wikipedia articles. <https://www.kaggle.com/datasets/disisbig/nepali-wikipedia-articles>.
URL <https://www.kaggle.com/datasets/disisbig/nepali-wikipedia-articles>
- Gupte, A., Romanov, A., Mantravadi, S., Banda, D., Liu, J., Khan, R., Meenal, L. R., Han, B., Srinivasan, S., 2021. Lights, Camera, Action! A Framework to Improve NLP Accuracy over OCR documents. arXiv.
URL <https://arxiv.org/abs/2108.02899>
- Hellwig, O., 2023. Indsenz OCR. ind.senz software, <http://www.indsenz.com/int/index.php> (accessed July 29, 2023).
URL <http://www.indsenz.com/int/index.php>
- Hendrycks, D., Gimpel, K., 2016. Gaussian error linear units (GELUs). arXiv preprint arXiv:1606.08415.
- Jawahar, C., Kumar, M. P., Kiran, S. R., 2003. A bilingual OCR for Hindi-Telugu documents and its applications. In: Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings. IEEE, IEEE, Edinburgh, UK, pp. 408–412.
URL <https://doi.org/10.1109/ICDAR.2003.1227699>

- Karayil, T., Ul-Hasan, A., Breuel, T. M., 2015. A segmentation-free approach for printed Devanagari script recognition. In: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on. IEEE, IEEE, Tunis, Tunisia, pp. 946–950.
URL <https://doi.org/10.1109/ICDAR.2015.7333901>
- Krishnan, P., Sankaran, N., Singh, A. K., Jawahar, C., 2014. Towards a Robust OCR System for Indic Scripts. In: Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on. IEEE, IEEE, Tours, France, pp. 141–145.
URL <https://doi.org/10.1109/DAS.2014.74>
- Kukich, K., dec 1992. Techniques for Automatically Correcting Words in Text. ACM Comput. Surv. 24 (4), 377–439.
URL <https://doi.org/10.1145/146370.146380>
- Kumar, A., Jun 2016. A Survey on Various OCR Errors. International Journal of Computer Applications 143 (4), 8–10.
URL <http://www.ijcaonline.org/archives/volume143/number4/25064-2016910142>
- Kundaikar, T., Pawar, J. D., 2020. Multi-font Devanagari text recognition using LSTM neural networks. In: First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019. Springer, Springer, JNU, Delhi, pp. 495–506.
URL https://doi.org/10.1007/978-981-15-0029-9_39
- Lehal, G., Singh, C., Lehal, R., 2001. A shape based post processor for Gurmukhi OCR. In: Proceedings of Sixth International Conference on Document Analysis and Recognition. IEEE, Seattle, WA, USA, pp. 1105–1109.
URL <https://doi.org/10.1109/ICDAR.2001.953957>
- Mathew, M., Jawahar, C., 2022. An empirical study of ctc based models for ocr of indian languages. arXiv preprint arXiv:2205.06740.
- Mathew, M., Singh, A. K., Jawahar, C., 2016. Multilingual OCR for indic scripts. In: 2016 12th IAPR Workshop on Document Analysis Systems (DAS). IEEE, IEEE, Santorini, Greece, pp. 186–191.
URL <https://doi.ieeecomputersociety.org/10.1109/DAS.2016.68>

Microsoft-corporation, (2019). Aparajita fonts. Accessed 24 January , 2019.

URL <https://catalog.monotype.com/font/microsoft-corporation/aparajita/regular>

Mohan, K., Jawahar, C. V., 2010. A Post-Processing Scheme for Malayalam Using Statistical Sub-Character Language Models. In: Proceedings of the 9th IAPR International Workshop on Document Analysis Systems. DAS '10. Association for Computing Machinery, New York, NY, USA, p. 493–500.

URL <https://doi.org/10.1145/1815330.1815394>

Nguyen, T. T. H., Jatowt, A., Coustaty, M., Doucet, A., jul 2021. Survey of Post-OCR Processing Approaches. ACM Comput. Surv. 54 (6).

URL <https://doi.org/10.1145/3453476>

OCROPY, 2023. Open Source Document Analysis and OCR Sys- tem. (accessed June 14, 2023).

URL <https://github.com/tmbdev/ocropy>

Pal, A., Mustafi, A., 2020. Vartani spellcheck – automatic context-sensitive spelling correction of ocr-generated hindi text using bert and levenshtein distance.

URL <https://arxiv.org/abs/2012.07652>

Pal, U., Chaudhuri, B. B., 1997. Printed devnagari script ocr system. VIVEK-BOMBAY- 10, 12–24.

Pal, U., Chaudhuri, B. B., 2004. Indian script character recognition: a survey. Pattern Recognition 37 (9), 1887–1899.

URL <https://www.sciencedirect.com/science/article/pii/S003132030400055X>

Pal, U., Kundu, P. K., Chaudhuri, B. B., 2000. OCR error correction of an inflectional indian language using morphological parsing. J. Inf. Sci. Eng. 16 (6), 903–922.

URL <https://api.semanticscholar.org/CorpusID:14740588>

Palit, S., Chaudhuri, B. B., 1995. A feature-based scheme for the machine recognition of printed Devanagari script. PP Das, BN Chatterjee (Eda.) Pattern Recognition, Image Processing and Computer Vision, Narosa Publishing House: New Delhi, India, 163–168.

Prahallad, K., Elluru, N. K., Keri, V., Rajendran, S., Black, A. W., 2012. The iiit-h indic speech databases. In: Interspeech. pp. 2546–2549.

Ramamoorthy, L., Choudhary, N., Varik, S., Tanawade, R. S., Gawas, Y. D., 2019. A Gold standard Konkani raw text corpus from Linguistic Data Consortium for Indian Languages. Accessed on May 19, 2023, <https://data.ldcil.org/text/text-raw-corpus/a-gold-standard-konkani-raw-text-corpus>.

URL <https://data.ldcil.org/text/text-raw-corpus/a-gold-standard-konkani-raw-text-corpus>

Rasagna, V., Kumar, A., Jawahar, C., Manmatha, R., 2009. Robust recognition of documents by fusing results of word clusters. In: 2009 10th International Conference on Document Analysis and Recognition. IEEE, IEEE, Barcelona, Spain, pp. 566–570.

URL <https://doi.org/10.1109/ICDAR.2009.135>

Rice, S. V., Nartker, T. A., 2022. The isri analytic tools for ocr evaluation. <https://github.com/eddieantonio/ocreval>.

URL <https://github.com/eddieantonio/ocreval>

Rigaud, C., Doucet, A., Coustaty, M., Moreux, J.-P., 2019. ICDAR 2019 Competition on Post-OCR Text Correction. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, Sydney, NSW, Australia, pp. 1588–1593.

URL <https://doi.org/10.1109/ICDAR.2017.232>

Saluja, R., Adiga, D., Chaudhuri, P., Ramakrishnan, G., Carman, M., 2017. Error Detection and Corrections in Indic OCR Using LSTMs. In: Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on. Vol. 1. IEEE, IEEE, Kyoto, Japan, pp. 17–22.

URL <https://doi.org/10.1109/ICDAR.2017.13>

Saluja, R., Punjabi, M., Carman, M., Ramakrishnan, G., Chaudhuri, P., 2019. Sub-word embeddings for OCR corrections in highly fusional indic languages. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, IEEE, Sydney, NSW, Australia, pp. 160–165.

URL <https://doi.org/10.1109/ICDAR.2019.00034>

Sankaran, N., Jawahar, C., 2012. Recognition of printed Devanagari text using BLSTM Neural Network. In: ICPR. Vol. 12. IEEE, Tsukuba, Japan, pp. 322–325.

URL <https://ieeexplore.ieee.org/document/6460137>

- Sankaran, N., Jawahar, C., 2013. Error Detection in Highly Inflectional Languages. In: 2013 12th International Conference on Document Analysis and Recognition. IEEE, Washington, DC, USA, pp. 1135–1139.
URL <https://doi.org/10.1109/ICDAR.2013.230>
- Sankaran, N., Neelappa, A., Jawahar, C., 2013. Devanagari text recognition: A transcription based formulation. In: 2013 12th International Conference on Document Analysis and Recognition. IEEE, pp. 678–682.
URL <https://doi.org/10.1109/ICDAR.2013.139>
- Shahi, T. B., Pant, A. K., 2018. Nepali news classification using Naïve Bayes, Support Vector Machines and Neural Networks. In: 2018 International Conference on Communication information and Computing Technology (ICCICT). "IEEE", Mumbai, India, pp. 1–5.
URL <https://doi.org/10.1109/ICCICT.2018.8325883>
- Shahi, T. B., Sitaula, C., 2022. Natural language processing for Nepali text: a review. Artificial Intelligence Review 55 (4), 1–29.
URL <https://doi.org/10.1007/s10462-021-10093-1>
- Shapiro, M. C., 1989. A primer of modern standard Hindi. Motilal Banarsidass Publisher, Bungalow road, Jawahar Nagar, Delhi 110007.
- Shivam, 2022. Hindi dataset.
URL https://huggingface.co/datasets/shivam/hindi_pib_processed
- Singh, V. N., Apr 2020. English to marathi translation.
URL <https://www.kaggle.com/datasets/vashistnarayansingh/english-to-marathi-translation>
- Sinha, R., 1973. A syntactic pattern analysis system and its application to devnagari script recognition. Ph.D. thesis, Electrical Engineering Department, indian Institute of Technology, Kanpur.
- Smith, R., sep 2007. An Overview of the Tesseract OCR Engine. In: 2007 9th International Conference on Document Analysis and Recognition. Vol. 2. IEEE Computer Society, Los Alamitos, CA, USA, pp. 629–633.
URL <https://doi.ieeecomputersociety.org/10.1109/ICDAR.2007.56>

Sorkin-Type, D., 2019. Dekko fonts. (accessed 24 January , 2019).

URL <https://fonts.google.com/specimen/Dekko>

Source, O., 2022. Tesseract OCR. Google, (accessed July 14, 2023).

URL <https://github.com/tesseract-ocr/>

Srigiri, S., Saha, S. K., 2020. Spelling Correction of OCR-Generated Hindi Text Using Word Embedding and Levenshtein Distance. In: Nath, V., Mandal, J. K. (Eds.), *Nanoelectronics, Circuits and Communication Systems*. Springer Singapore, Singapore, pp. 415–424.

URL https://doi.org/10.1007/978-981-15-2854-5_36

Sun, C., Qiu, X., Xu, Y., Huang, X., 2019. How to Fine-Tune BERT for Text Classification? In: *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*. Springer, pp. 194–206.

URL https://doi.org/10.1007/978-3-030-32381-3_16

tdil, 2018. eAksharaya. TDIL, accessed August 14, 2018).

URL <http://tdil-dc.in/eocr/index.html>

TDIL, 2019a. Hindi-Marathi text corpus. Technology Development for Indian Languages Programme, MeitY, Govt of India., (accessed 26 January , 2019).

URL <http://tdil.meity.gov.in/>

TDIL, 2019b. Konkani Monolingual Text Corpus ILCI-II. Technology Development for Indian Languages Programme, MeitY, Govt of India., (accessed 26 January , 2019).

URL <http://tdil.meity.gov.in/>

TDIL, 2019c. Nepali General Text Corpus. Technology Development for Indian Languages Programme, MeitY, Govt of India., (accessed 26 January , 2019).

URL <http://tdil.meity.gov.in/>

Vinitha, V., Mathew, M., Jawahar, C., 2017. An Empirical Study of Effectiveness of Post-processing in Indic Scripts. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 7. IEEE, IEEE, Kyoto, Japan, pp. 32–36.

URL <https://doi.org/10.1109/ICDAR.2017.362>

Vinitha, V. S., Jawahar, C. V., 2016. Error Detection in Indic OCRs. In: 2016 12th IAPR Workshop on Document Analysis Systems (DAS). IEEE, Santorini, Greece, pp. 180–185.

URL <https://doi.org/10.1109/DAS.2016.31>

List of Publications/Presentations

International Journals

1. **Kundaikar, T.**, Fadte, S., Karmali, R., Pawar, J.D. (2024). “Automatic Hindi OCR error correction using MLM-BERT.” *Ingénierie des Systèmes d’Information*, Vol. 29, No. 2, pp. 619-626. <https://doi.org/10.18280/isi.290223>
2. “Multi-font OCR for low-resource language Konkani using Tesseract LSTM” *SN computer Science*, SpringerNature, ISSN:2662-995X, E-ISSN:2661-8907, (under review).
3. “Post-OCR Error Detection for Devanagari” *Language Resources and Evaluation*, Springer-Nature, ISSN:1574-020X, E-ISSN: 1574-0218 (under review).
4. “MB-NSC:A Hybrid approach for Automatic Error Correction of Konkani OCRred documents using MLM-BERT and Norvig Spell Corrector”, *Sāadhanā*, Indian Academy of Sciences and SpringerNature, E-ISSN 0973-7677, (under review)

International Conferences

1. **Kundaikar, T.**, Pawar, J.D. (2020). “Multi-font Devanagari Text Recognition Using LSTM Neural Networks.” In: Luhach, A., Kosa, J., Poonia, R., Gao, XZ., Singh, D. (eds) *First International Conference on Sustainable Technologies for Computational Intelligence. Advances in Intelligent Systems and Computing*, vol 1045. Springer, Singapore. https://doi.org/10.1007/978-981-15-0029-9_39

National Conferences

1. **Kundaikar, T.**, Pawar, J.D. (2019). “Multi Font Error Correction for Devanagari Script in OCR”, presented at *SOIL-Tech 2019: Sanskrit and other Indian Languages-Technology* held in the School of Sanskrit and Indic Studies JNU, New Delhi held on 15th February 2019 to 17th February 2019.

Index

- accent mark, 21
- accuracy, 2, 7, 14, 17, 18, 23, 55–57, 59, 63, 69–71, 78, 79, 81, 84, 85, 89–91, 97, 99, 102, 103, 109, 117, 119–121, 125, 127, 131, 135, 138, 143, 144, 153, 174
- activation, 69, 70, 81
- additional consonant, 33, 34
- algorithm, 9, 55, 60–63, 84, 97, 101, 148
- alignment, 45
- Amiko, 51, 129, 131, 133, 135–137, 175, 176, 178, 180, 182
- analysis, 18, 34, 63, 71, 109, 120, 125
- angular terminal, 4
- Anusvara, 96
- anusvara, 43, 49, 72, 86, 97, 139
- Aparajita, 4, 5, 9, 11, 42, 44, 51, 54, 57, 59, 63, 98, 117, 118, 120, 121, 123, 125, 129–131, 137, 139, 148, 150, 153
- appendix, 155, 167
- architecture, 78, 80, 102, 103, 109, 120
- Asar, 51, 129–131, 137, 175, 176, 178, 180, 182
- Assamese, 16, 21
- automation, 44
- axis, 4
- bag-of-words, 19
- Baloo, 4, 51, 117, 118, 121, 122, 125, 129–131, 137, 139, 175, 176, 178, 180, 182
- Bangla, 16, 18
- Bays theorem, 83
- Bengali, 16, 21
- BERT, 68, 81, 102, 103, 105, 107–109
- bigram, 67
- bilingual, 14
- Biryani, 4, 51, 117, 118, 121, 122, 125, 129–131, 135, 137, 152, 175, 176, 178, 180, 182
- BLSTM, 16
- Bodo, 21, 23
- borrowed-word, 26, 140, 143
- Brahmi, 1
- Cambay, 5, 23, 51, 129–131, 135, 137, 175, 176, 178, 180, 182
- candra, 43, 49, 86, 97, 140, 141
- candrabindu, 86, 139
- canonical, 114
- canonical equivalence, 26
- CER, 16, 56, 84, 99, 137
- character, 8, 11, 18, 21, 22, 24, 35, 38, 40, 45, 60, 85, 86, 97, 121, 137, 147
- checkpoint, 134

classification, 3
 cluster, 18
 CNN, 102, 103, 105, 109
 code-mix, 110, 115
 code-mixing, 103
 command, 134
 comparative, 22
 conclusion, 12, 39, 41, 54, 98, 114, 143
 confusion-matrix, 58
 conjunct, 1, 4, 5, 8, 23, 49, 72, 87, 137, 140
 conjunct-consonant, 5, 21, 36, 38, 43, 131
 consonant, 1, 5, 8, 18, 21, 25, 26, 30, 31,
 33, 36, 37, 43, 48, 72, 85, 91, 131,
 137, 139, 140, 144, 153, 159
 consonant-cluster, 21, 36
 context, 19, 21, 36, 56, 63, 67, 68, 72, 75,
 79, 90, 92, 94, 95, 98, 110, 151
 contextual, 18
 contrast, 4
 contribution, 147
 corpus, 2, 7, 23, 24, 26, 28, 32–34, 47–49,
 52, 69, 70, 80, 81, 86, 88, 97, 108,
 114, 127, 129
 Danda, 35
 data, 22, 23, 43, 45, 97, 118, 128, 142
 dataset, 1, 5, 8, 11, 16, 24, 41, 44, 46, 48,
 51–54, 56, 57, 59, 68, 71, 79, 80,
 85, 90, 104, 107, 118, 121–123,
 125, 129, 131, 132, 135, 137, 141,
 152, 176, 178
 dead-consonant, 22, 36
 deep-learning, 14, 17, 19, 55
 Dekko, 4, 117, 118, 121, 122, 125, 152
 deletion, 7, 57, 59, 63, 67, 84, 85, 99, 117,
 120–123, 125, 135, 137, 153
 dependant-vowel, 137, 139
 detection, 103
 Devanagari, 1, 4, 5, 7, 8, 11, 14, 16, 21–23,
 25, 26, 28, 32, 36, 38, 39, 43, 52,
 53, 55, 79, 85, 87, 97, 99, 102, 105,
 109, 110, 115, 127, 129–131, 134,
 136, 137, 139, 141, 142, 144, 147,
 152, 153
 diacritic, 21, 30, 36
 dictionary, 17, 18, 56, 58, 68, 69
 discussion, 38, 134
 Dogri, 21, 23
 domain, 103, 115
 dropout, 69, 70, 81
 DTW, 18
 Eczar, 51, 129, 130, 137
 edge, 18
 eekar, 43, 48
 ekar, 72
 embedding, 69, 70, 81
 empirical, 8, 147, 153
 engine, 134
 English, 1, 28, 49, 81, 97, 114, 115
 epoch, 69, 70
 error, 85, 88, 94, 122, 123, 125, 137–139,
 141, 167, 173, 180
 error-analysis, 121, 136, 144, 153
 error-correction, 2, 3, 9, 11, 17, 18, 45, 55,
 58, 61, 67, 68, 71, 72, 75, 78, 81,

82, 90, 91, 97–99, 102, 117, 148,
151

error-detection, 3, 11, 53, 56, 60, 71,
101–103, 105, 109, 110, 112, 153

error-mapping, 60, 61

evaluation, 56, 57, 71, 85, 99, 121, 135

experiment, 63, 120, 132

F-score, 102, 108, 110

fine-tuning, 129

FN, 110

font, 1, 3–5, 23, 41, 44, 50–52, 56, 57, 59,
118, 120, 121, 123, 125, 127, 130,
131, 133, 135, 136, 142, 152, 176,
178, 180, 182

foreign word, 18, 67

FP, 110

French, 1

frequency, 22, 24–26, 28, 32, 35–37, 58,
60–63, 85, 167, 173

function, 69, 70, 81

future work, 12, 153

Gargi, 51, 129–131, 137, 175, 176, 178,
180, 182

GELU, 69, 70, 81

German, 1

glyph, 3

Goa, 127

gradient, 69, 70, 81

graph-based, 18

grey value, 4

ground-truth, 1, 45

Gujarati, 21, 85, 102

Gurumukhi, 16, 18

halant, 36, 87, 89

Hindi, 7, 11, 14, 16, 18, 21–23, 25, 26, 28,
30, 32–41, 45, 46, 50, 52–55, 71,
72, 78, 79, 84, 85, 90, 98, 102–104,
108, 110, 115, 117, 125, 127, 129,
131, 140, 151–153, 155, 159, 160,
165, 174

horizontal line, 36

horizontal terminal, 4

hybrid, 75, 78, 84

inclination, 4

indic, 52

inflectional, 7, 17, 18

insertion, 7, 57, 59, 63, 67, 84, 85, 99, 117,
120–123, 125, 135, 137, 153

inter-language, 102

interjection, 114

intersection, 18

introduction, 21, 101, 127

Jaccard-similarity, 22, 38–40

Kannada, 2, 16, 18, 102

Kerala, 2

Konkani, 1, 2, 7, 11, 16, 17, 21–26, 28,
30–32, 34–41, 46–49, 52, 54, 56,
78–82, 84, 85, 89, 90, 92, 97, 98,
102–104, 108, 110, 115, 127, 129,
131, 132, 134–137, 140–144,
151–153, 155, 159, 160, 165,
174–176, 178, 180

learning-rate, 120
 Levenshtein distance, 2, 19, 55, 69, 82
 lexicon, 17
 ligature, 4, 8, 21–25, 36–38, 88, 127, 160, 165, 166
 likelihood, 82
 linguistic, 21
 list, 63
 Lohit-Devanagari, 5, 23, 51, 129, 131, 135, 137, 175, 176, 178, 180, 182
 loss, 70, 109
 low-resourced, 2, 7, 102, 103, 115, 127, 129, 144, 152, 153
 LSH, 18
 LSTM, 2, 7, 16, 19, 55, 120, 125, 132–134, 144, 152, 153
 Maithili, 21, 23
 Malayalam, 2, 16, 18, 21, 102
 Manipuri, 21
 Marathi, 11, 21–26, 28, 30–41, 52, 54, 55, 85, 102–104, 108, 115, 127, 152, 153, 155, 159, 160, 165
 mask, 108
 matra, 43, 86
 MB-NSC, 10, 11, 56, 75, 78–80, 84, 90, 92, 93, 95–97, 149, 153
 mBERT, 10, 11, 149, 153
 metric, 56, 71, 135
 misclassification, 102
 MLM, 105
 MLM-BERT, 9, 11, 56, 67, 68, 75, 79, 81, 82, 91, 92, 97, 98, 149, 151
 model, 53, 67, 69, 70, 72, 75, 81, 91, 92, 102, 105, 107, 112, 120, 128, 129, 132, 134, 140
 modifier, 49, 72, 89
 monolingual, 52
 morph-analyzer, 17
 morphology, 17
 motivation, 5, 23, 76
 multi-consonant, 22
 multi-consonant conjunct, 36
 multi-font, 5, 9, 12, 16, 51, 56, 117, 120, 123, 125, 127, 131, 134–138, 141–144, 148, 152, 153, 175, 176, 178, 180
 multilingual, 16, 108, 109
 n-gram, 17
 Nakula, 4, 9, 11, 42, 44, 50–52, 54, 57, 59, 63, 98, 117–121, 125, 129, 131, 135, 137, 148, 150, 153, 175, 176, 178, 180, 182
 nasal, 43
 Nepali, 11, 21–26, 28, 30–41, 52, 54, 102–104, 108, 114, 115, 127, 152, 153, 155, 159, 160, 165
 NER, 17
 network, 14
 neural, 14
 NLP, 2, 7, 127, 144, 153
 node, 18
 NSP, 105
 nukta, 22, 34, 36, 87
 number, 24, 95, 114, 140

numeral, 43, 48, 87
 numeric set, 60

 objective, 78
 observation, 38
 OCR, 1, 2, 5, 7, 11, 13, 14, 16, 18, 23, 41,
 44–46, 51, 52, 54–57, 59, 60, 63,
 71, 72, 78, 79, 84, 85, 88, 90, 92,
 98, 99, 102–105, 107, 115, 117,
 119–123, 125, 127, 129–132,
 134–137, 141–143, 151–153, 174,
 175, 180
 ookar, 43, 49, 72
 open-source, 127
 optimal-path, 18
 optimization, 70, 81
 orthographic, 21

 performance, 11, 41, 57, 59, 79, 84, 104,
 108, 117, 119, 130, 131, 134, 135,
 142, 153
 POS, 17
 post-OCR, 54
 post-processing, 3, 7, 8, 11, 13, 16–18, 41,
 44–46, 51, 54–56, 67, 99, 102,
 120, 140, 144, 148, 151, 153
 pre-processing, 3, 24
 pre-trained, 68, 81, 107, 108, 134
 precision, 110
 probability, 69, 82, 84, 92
 procedure, 44, 63, 132
 pronunciation, 21, 36
 proposed solution, 132

 punctuation mark, 21, 25, 35, 36, 72, 112
 Punjabi, 16, 18

 quotation mark, 24, 86, 139–141

 rate, 69, 70, 81, 89
 rebuild, 134
 recall, 110
 recognition, 14
 recommendation, 72
 recursive, 45
 related work, 14
 Research contribution, 8
 research gaps, 19
 research-gaps, 13
 result, 63, 71, 72, 109, 120, 134
 review, 11, 14
 RNN, 16, 19, 102, 120
 roman, 2, 14
 root, 18

 Sanskrit, 18, 21, 23, 26, 32, 35, 55, 102
 Santhali, 21, 23
 score, 39, 40
 script, 1, 14, 16–18, 21, 26, 32, 36, 52, 55,
 79, 102, 128–130, 144, 153
 segmentation, 3, 14, 86
 segmentation-based, 48
 segmentation-free, 14
 semantic, 18
 semantic-rule, 17
 shape-based, 18
 similarity, 23, 39, 40
 Sindhi, 21, 23

SLM, 18
 spell-correction, 19, 56, 98, 151
 standard, 53
 structure, 132
 study, 40, 41, 153
 sub-character, 18
 sub-word, 68, 107
 sub-word embedding, 18
 substitution, 7, 57, 59, 63, 85, 99, 120–123, 135, 137, 141, 153, 167, 173, 180
 sugestion, 63
 summary, 147
 syllable, 8, 11, 21–26, 32, 36, 38, 40, 43, 48, 58, 60–63, 85, 87, 88, 127, 137, 140, 147, 153, 165, 167, 173, 175
 syllable-based, 56
 syntactic, 18
 tagger, 17
 Tamil, 16, 21, 85
 Telugu, 14, 16, 18, 21
 terminology, 58
 test, 46, 53, 72, 79, 80, 121–123, 129, 135
 test-dataset, 52, 63
 Text recognizer, 117
 text-recognition, 3
 TN, 110
 token, 68, 81, 108
 tokenization, 24
 tokenizer, 25, 68, 69, 81, 105, 107, 108
 tool, 17, 56, 79, 85, 90, 102, 103, 120, 121, 127, 143
 TP, 110
 train, 53, 70, 72, 80, 97, 105, 108, 129
 training, 128, 132, 134
 transcription, 128
 transliteration, 26, 103, 114, 115, 180
 TrueType, 4
 Tulu, 85
 tuple, 24, 25, 59–61
 typeface, 3
 unicode, 4, 86
 validation, 107, 108
 vedic tone, 21, 35
 vertical conjunct, 23
 vertical stem, 4
 vertical terminal, 4
 viram, 21, 22, 36, 87
 vocabulary, 103
 vowel, 1, 5, 21, 25, 26, 32, 33, 36–40, 43, 48, 86, 87, 140, 155, 160
 weight, 58, 63
 WER, 16, 56, 71, 72, 84, 91
 word-embedding, 19, 55
 WSSC, 9, 11, 56, 58, 60, 61, 63, 98, 149, 153
 ZWJ, 88, 89, 97, 99
 ZWNJ, 88