Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Enhancing Sindhi (Devanagari) OCR Performance Through MLM-BERT-Based Error Correction Model

Arvind Kaur, Gurpreet Singh Lehal

Department of Computer Science, Punjabi University, Patiala

Optical Character Recognition (OCR) systems often face challenges in accurately processing text in morphologically rich Indic scripts like Devanagari, which is used by languages such as Hindi, Marathi, Sanskrit, and Sindhi. The complexity of these languages, where words can change form based on gender, meaning, or context, makes consistent text recognition difficult. While OCR systems have been developed for various languages, the Sindhi language written in Devanagari script has not been extensively studied, particularly in terms of OCR and post-processing. This work focuses on improving Sindhi (Devanagari) OCR accuracy by introducing a post-processing model based on Masked Language Modeling with BERT (MLM-BERT). The performance of the trained MLM-BERT model was evaluated on two distinct testing datasets: one from the same domain and the other from a different domain. The model improved OCR accuracy by 4.01% on the same-domain dataset and 1.90% on the different-domain dataset, demonstrating its effectiveness in enhancing OCR accuracy across varying contexts.
Keywords: Sindhi (Devanagari) Script, MLM-BERT, OCR, Natural Language Processing, Error Correction, Deep Learning, Post-processing etc.

1. Introduction

The Sindhi language, written in the Devanagari script, belongs to the Indo-Aryan language family. The Sindhi Devanagari script is an adapted version of the standard Devanagari script which captures the distinct sounds and linguistic features of Sindhi. Devanagari's adoption for Sindhi transcription gained prominence due to its compatibility and widespread use. This script features unique letter forms and ligatures, which are created by connecting various characters to form words. These ligatures and connections not only contribute to the script's distinctive visual identity but also highlight its significance in Sindhi cultural and linguistic history. In addition to the consonants found in the Devanagari script for Hindi, Sindhi (Devanagari) includes four additional consonants, which are formed by adding a diacritical

bar beneath the standard consonants represented as ग़, ज़, ड़ and ब़. These consonants are unique to Sindhi and have distinct phonetic values compared to GA, JA, DA, and BA [1]. Therefore, developing effective Optical Character Recognition (OCR) solutions for the Sindhi Devanagari script is a significant area of research. OCR involves the extraction of text from digitally scanned documents, and it is essential to maintain the meaning and integrity of the text for use in natural language processing (NLP) applications. NLP is vital for the preservation, processing, and expansion of language use in the digital era. However, the Devanagari script presents several challenges for OCR systems, including its complexity such as the formation of intricate syllables from combinations of vowels, consonants, and conjunct consonants and potential errors in contextual recognition. Moreover, the scarcity of language resources further complicates the training of OCR and NLP systems. Technological advancements are mainly concentrated on languages with abundant data, leaving many other languages overlooked.

In the case of the Sindhi language written in Devanagari script, there is currently no publicly available OCR solution. To address this gap, we have developed a Deep Learning CNN-BLSTM model specifically for the Sindhi (Devanagari) script. The model has achieved an accuracy of 94.46% without the application of post-processing techniques. To enhance the OCR output further, it is essential to implement additional post-processing methods, which can significantly improve the accuracy.

Our key contributions are outlined as follows:

- A benchmark dataset specifically designed for OCR post-correction of the Sindhi language written in the Devanagari script is presented.
- An OCR post-correction approach utilizing the MLM-BERT model for the Sindhi (Devanagari) script, achieving a significant improvement in accuracy is introduced.

2. Related Work

Recent post-processing techniques aimed at enhancing OCR accuracy include dictionary-based methods [2], statistical language models [2, 3], deep learning with LSTM [4], word embedding combined with Levenshtein distance [5], sub-word embedding [6] and n-grams [7]. However, these techniques often fall short in addressing contextual errors. To overcome this limitation, we propose a context-sensitive automatic error correction method designed to improve the accuracy of Sindhi Devanagari OCR output. The objective of this approach is to correct errors in text documents generated by the Sindhi Devanagari OCR system by providing suggestions for incorrect words based on their context, thereby enhancing overall accuracy. Error correction is essential for enhancing the accuracy of OCR-generated text. This section explores various post-processing approaches used to correct OCR errors in Indian languages. [8] have briefly investigated a range of post-processing techniques. The post-processing methods are categorized into manual and semi-automatic approaches and further classified into isolated-word and context-dependent types based on the level of information utilized. Some methods identify the best possible alternative, while others generate the top-n potential alternatives.

The Dictionary-based approach on Hindi OCR showed an accuracy of 93% [9]. The classification output undergoes error detection and correction by partitioning the dictionary for faster processing. The correction process involves three steps: selecting a relevant dictionary partition based on the input word's characteristics, matching the word with selected candidates, and either confirming the word's accuracy if found or seeking the closest match using a distance measure or by generating aliases for an exact match if the word is not in the dictionary. The issue with the dictionary-based approach is that it automatically considers a word correct if it exists in the dictionary, and it lacks the ability to correct contextual errors.

This paper [10] presents an OCR error detection and correction method for Bangla, a highly inflectional Indian language. The approach uses morphological parsing with separate lexicons for root words and suffixes to identify and correct errors. The system achieves an 84.22% success rate in suggesting the correct word by generating alternatives and testing grammatical agreement.

A shape-based post-processing system for OCR of Gurmukhi script has been developed, dividing Punjabi corpora by word size and shape [11]. The system combines statistical analysis of syllable combinations, corpora lookup, and recognition of common words. It encodes input words based on shape similarity, matching them with dictionary entries or suggesting structurally similar alternatives. This method improves recognition rates from 94.35% to 97.34% but struggles with similar characters.

Locality Sensitive Hashing (LSH) [12] was used to cluster words for enhancing Telugu OCR accuracy, achieving 79.12%. OCR outputs within each cluster were improved using Character Majority Voting or Dynamic Time Warping. However, the technique struggles with unique words that appear only once in a cluster.

A post-processing method that uses sub-character level statistical language models to improve word recognition is presented [13]. The technique models the recognition task as an optimization problem using a multi-stage graph, where edges encode language data and nodes capture visual similarities. Tested on over 10,000 Malayalam words, it achieves 95% accuracy but struggles with rare words, proper nouns, and foreign terms.

A straightforward method for learning word representations by integrating subword information through character n-grams into the skip-gram model is also explored [14]. The approach, is fast to train and requires no preprocessing or supervision. They have demonstrated that it outperforms models that ignore subword information and those relying on morphological analysis. Indic languages often contain many out-of-vocabulary (OOV) words due to complex word fusion rules, which complicates OCR error correction. Sub-word units, such as n-grams, can be extracted from OCR and language texts to capture context and detect errors, especially those related to word conjoining rules. This study explores two encoding methods for enhancing LSTM-based OCR correction models: one using sub-word frequency values for faster convergence and improved accuracy, and another using trainable sub-word embeddings, leading to significant gains in F-Scores and word-level accuracy across four languages. Sub-word embeddings have been successfully applied to OCR error correction in Hindi, Sanskrit, Kannada, and Malayalam, achieving a 90.42% word accuracy for Hindi.

OCR systems for Indian languages like Hindi face accuracy issues due to diverse characters and spelling errors. A Hindi spelling correction system using neural word embeddings and Levenshtein distance is proposed [5]. The Continuous Bag-of-Words (CBOW) model generates word embeddings from large corpora, while dictionary and context analysis detect errors. Candidate corrections are generated based on embedding similarity and evaluated using Levenshtein distance. Although the system achieves reasonable accuracy on 'The Gita' it struggles with contextual suggestions due to not considering word order.

An automatic model for OCR error correction using correction pattern edits and an evolutionary algorithm is presented [15]. By combining these with a variant of the self-organizing migrating algorithm and a fitness function based on linguistic features, the model significantly improves candidate generation and error correction. It achieves a 33.7% improvement in Levenshtein distance and outperforms many baseline methods in the ICDAR 2017 competition, though it still lags behind the top statistical and neural machine translation models. Future work aims to address additional OCR error types and refine the approach further.

N-gram counts for error detection are introduced, which simplifies the process and reduces computational costs. By focusing on uni-grams, bi-grams, and tri-grams, the method achieves state-of-the-art F1-scores for eight out of ten European languages in the ICDAR 2019 competition, outperforming previous systems. The most significant improvement was observed in Spanish, where the F1-score increased from 0.69 to 0.90, while the smallest gain was in Polish, with an improvement from 0.82 to 0.84. This approach's simplicity eliminates the need for complex feature engineering and proves effective with relatively small datasets [7].

[4] used an LSTM-based character-level language model with a fixed delay to handle both error detection and correction in Indic OCR. It avoids suggesting corrections for correctly recognized words. Extensive testing on four Indian languages shows FScores above 92.4% and a reduction in Word Error Rates by at least 26.7%.

3. Motivation

Recent literature surveys reveal that no research has been conducted on post-processing the Sindhi (Devanagari) text output from Optical Character Recognition (OCR). The review of existing studies suggests that post-processing plays a vital role in improving the performance of Hindi OCR. However, the most widely used methods fail to address context-sensitive error correction, as demonstrated by the three examples in Table 1. The text highlighted in yellow represents a word that is technically correct according to the language dictionary but is incorrect when considering the context. These types of contextual errors must be accurately handled by the error correction model. To address such cases, an approach is needed that can suggest words based on the surrounding context.

This can be accomplished by utilizing advanced techniques like the Masked Language Model (MLM) with Bidirectional Encoder Representations from Transformers (BERT), which have not yet been widely explored for OCR error correction in Indian languages.

Table 1: Few examples of OCR-recognized errors that necessitate contextual suggestions

Example 1	
Actual Text(Sindhi Devanagari)	मैत्री महाविद्यालय में पढ़ाई कई हुआईं ।
Translation (Hindi)	उन्होंने एक कॉलेज में पढ़ाई की थी।
Translation (English)	He had studied in a college.
OCR Recognized Text	मैत्री महाविद्यालय में पढ़ाई कई हुआ ।
Translation (Hindi)	मैत्री ने एक कॉलेज में पढ़ाई की थी।
Translation (English)	Maitri studied in a college.
Example 2	
Actual Text(Sindhi Devanagari)	क़ाज़ी कादन जे बैतनि जी बोली सिराइकी मुलतानी आमीज़श वारी ऐं पंजाबीअ खां मुतासिर सिंधी बोली आहे ।
Translation (Hindi)	काजी खान के सितार की भाषा सिरिलिक माल्टीज़-प्रेरित और पंजाबी-प्रेरित सिंधी बोली है।
Translation (English)	The language of the sitar of Kaji Khan is Cyrillic Maltese-inspired and Punjabi-inspired Sindhi dialect.
OCR Recognized Text	क़ाज़ी कादन जे बैतनि जी बोली सिराइकी मुलतानी आमीज़श वारी ऐं पंजाबीअ खां मुतासिर सिंध बोली आहे ।
Translation (Hindi)	काजी खान के सितार की भाषा सिरिलिक माल्टीज़-प्रेरित और पंजाबी-प्रेरित सिंधी है।
Translation (English)	The language of Kaji Khan's sitar is Cyrillic Maltese-inspired and Punjabi-inspired Sindhi.

The MLM-BERT model has been applied to tasks like detection and correction, showcasing its capability to streamline the model's architecture while delivering contextually relevant suggestions [16-20]. Also, the MLM-BERT model for correction of Hindi text has been explored in [21]. They have presented an OCR error correction method that leverages an advanced Masked Language Model (MLM) with BERT. Specifically, the "*bert-base uncased* " model, pre-trained on uncased English text, was used for further training. By masking words that contain errors, the MLM BERT model generates suggestions for the masked word based on the surrounding context. This approach achieved a 3.58% improvement in word accuracy compared to Tesseract OCR.

To develop an MLM BERT model for Sindhi (Devanagari) script, a BERT tokenizer and MLM BERT model for Sindhi (Devanagari) are required. This paper explores an OCR error correction method that is distinct from any pre-existing trained BERT models like "*bert-base uncased* " as used in [21], to offer contextually aware corrections. The process used is known as "pre-training from scratch". In the absence of pre-existing models for a language, a new BERT model has been initialized with random weights and trained on a large dataset in Sindhi (Devanagari) script. During this training, the model acquires an understanding of the language's syntax, semantics, and contextual relationships directly from the data, enabling it to perform tasks such as error correction in natural language processing with high effectiveness

once sufficiently trained.

4. Proposed Methodology

This section outlines two key components of the experimental setup. The first component explains the specifics of training dataset and the second component explains the process of creating the test dataset for the error correction model.

4.1 Creation of Training Dataset

The dataset for the training of proposed error correction model comprises of 62,088 sentences with varying lengths. The categories of Devanagari words [22] found in this training dataset are presented below:

Words without modifier: Words with Devanagari vowels or consonants with no modifiers, e.g. कछ, मन, उन, इन, करण

Words with Matra: Words with matras which is a dependent vowel sign like ◌े, ◌ै, ◌ो, ◌ौ, e.g. जे, हो, जेके, दौर, पैर, घणो

Words with Eekar: Words with modifier which is a dependent vowel sign like ◌ि, ◌ी, e.g. अथिम, विधि, लिखण, अहमियत, हकीम

Words with Ookar: Words with modifier which is a dependent vowel sign like ◌ु, ◌ू, e.g. जुदा, हुन, मूजुबु, दूबदू, पुथल

Words with conjunct consonant: Words which are formed when one consonant is followed by another consonant and both are written together as a single unit e.g.व्याख्या, दस्तयाब, कैप्टन, प्रान्त, वक्त

Words with Anusvara: Words with symbol ◌ं e.g. जां, सिंधी, जंहिं, हितां, पंहिंजे

Words with Chandrabindu: Words with symbol ◌ँ e.g. गाँधीअ, गाँधीधाम, दांहँ, नींहुँ, मुंहँ

Words with Nukta: Words with symbol ◌़ e.g. ओताक़, निचोइण, अलहाज, पैग़ाम, ब़दन

Words with Sindhi (Devanagari) characters: Words with characters specific to Sindhi (Devanagari script) like ग़, ज़, ड़, ब़ e.g. ड़हर, ब़ियनि, ज़ातलु, सुग़र्वासी, अग़ियाड़ीअ

4.2 Creation of Test Dataset

The testing dataset is needed to evaluate the performance of proposed Sindhi (Devanagari) trained MLM-BERT model. This section outlines the process of creation of two Test datasets to assess the model's performance which is described below:

1. Due to the absence of a publicly available OCR dataset for the Sindhi (Devanagari) script, we developed a custom OCR system by training a CNN-LSTM model on 111,010 images paired with their corresponding text lines. This system achieved an accuracy of 94.46% on test dataset.
2. The trained OCR system was subsequently used to generate test dataset for evaluating the error correction model. The OCR model takes Bitmap images as input and produces a set of corresponding text lines, referred to as N_OCR.
3. To create the dataset, Ground Truth lines (N_GT) need to be aligned with the OCR output (N_OCR) to identify incorrect words in the sentences. For word-level

alignment, the Recursive Text Alignment System (RETAS) [23] is used. In this process, each word in the Ground Truth lines is compared with the corresponding OCR-generated word. If the words match, the OCR word is labeled as 0; if they do not, it is labeled as 1.

4. The *Sindhi_Testdataset_A* consists of 400 pairs of N_GT and N_OCR lines, totaling 7,094 words. The N_GT lines were extracted from Sindhi (Devanagari) text within the same domain as the one used for model training. In contrast, the second dataset, *Sindhi_Testdataset_B*, also includes 400 N_GT and N_OCR lines, containing 8,750 words, but is derived from Sindhi (Devanagari) text belonging to a completely different domain than the one used for training.

5. Framework for Error Correction Experimentation

BERT is engineered to grasp the context of a word within sentences by simultaneously considering the words that precede and follow it, making it bidirectional. Unlike earlier models that process text in only one direction, either left-to-right or right-to-left, BERT analyzes text in both directions. This dual approach enables BERT to capture context more accurately. Built on the Transformer architecture, BERT leverages a self- attention mechanism that allows the model to evaluate the significance of each word in a sentence relative to the others, thereby enhancing its understanding of context.

MLM, or Masked Language Model, is a crucial pre-training task in BERT. During this process, certain words in a sentence are randomly masked, replaced with a [MASK] token. BERT is then challenged to predict the original word by analyzing the surrounding context, enabling it to learn deep contextual relationships within the text [24].

In the proposed error correction method, we first mask the incorrect word and then use the model to suggest a contextually appropriate replacement for the masked word. Figure 1 illustrates the proposed system, which is composed of three key modules: 1) Training the BERT tokenizer, 2) Training the BERT Masked Language Model (MLM) model, and 3) Performing Error Correction using the MLM BERT model. A detailed explanation for each phase of the process is described below:

1. **Training the BERT tokenizer:** The BERT tokenizer is a key component of the BERT (Bidirectional Encoder Representations from Transformers) model, designed to convert text into a format suitable for processing by the model. In the proposed method, the 'BertWordPieceTokenizer' is utilized to break down words into subword units and assign each token a unique ID. When the BERT tokenizer processes a word, it splits it into two subwords or tokens. The first token is typically a common prefix found in the corpus, while the second token, which represents the suffix, is pre-fixed with two hashes to indicate that it follows another subword [25]. Since the BERT model requires input sentences of a fixed length, padding is applied to ensure all sentences are of equal length before being fed into the model. The tokenizer creates an attention mask, a binary mask with 0s and 1s, where 1 indicates the tokens that the model should focus on, and 0 represents padding that should be ignored. This allows the model to concentrate on the actual content while disregarding the padding.

The tokenizer 'Sindhi-Dev Tokenizer' then produces a list of token IDs and the attention mask, both of which are input into the BERT model for further processing.

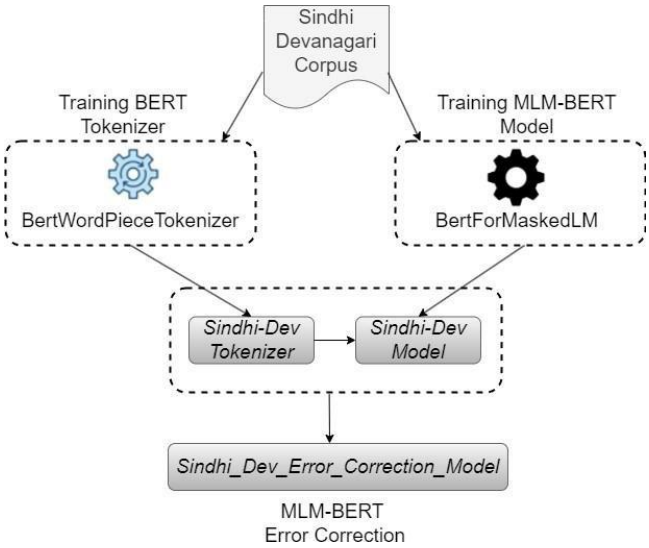


Figure 1: Overall Research Methodology Process

2. **Training the BERT Masked Language Model (MLM) model:** The 'BertWordPieceTokenizer' is employed to manage subword tokenization. This tokenizer is trained on the provided text files to create a vocabulary of 30,522 tokens, including special tokens. It is also set to truncate sequences to a maximum of 512 tokens. Once trained, the tokenizer is reloaded using 'BertTokenizerFast', which is used for encoding and decoding text during model training and evaluation. Additionally, a custom data collator is implemented to handle token masking for Masked Language Modeling (MLM) during training. This collator randomly masks certain tokens, substitutes some with others, and tracks these modifications for accurate loss computation. The model used for training is 'BertForMaskedLM', a variant of BERT specifically designed for masked language modeling tasks. This model consists of 12 layers, 768 hidden units, 12 attention heads, and a total of 110 million parameters. The training parameters were configured as follows: Gaussian Error Linear Unit (GELU) was used as the hidden activation function, with the maximum position embeddings capped at 512. The training process was conducted for 26 epochs, utilizing a per device batch size of 8. Gradient accumulation steps were set to 8, with a dataset containing 62,088 sentences, and the total number of optimization steps were 15,000. After the completion of training, the training loss came out to be 0.072. The trained MLM-Bert model is named as 'Sindhi-Dev Model'.

3. **Performing Error Correction using the MLM BERT model:** In this module, incorrect words in a sentence are masked using the [MASK] token. The word

correction process leverages the 'Sindhi-Dev Model' along with its corresponding tokenizer, 'Sindhi-Dev Tokenizer'. The input OCR text with the masked token is first tokenized using the 'Sindhi-Dev Tokenizer', generating token IDs. The 'Sindhi-Dev Model' then predicts potential replacements for the masked word, providing token IDs and their associated probabilities. These token IDs are converted back to words using the 'Sindhi-Dev Tokenizer'. The incorrect word is typically replaced with the prediction that has the highest probability. While this method is highly effective at selecting contextually appropriate words, it may sometimes introduce words that, though correct in context, are incorrect for the specific sentence content. For example, if the error correction model is trained on Sindhi Devanagari lines (as shown in Table 2), and the sentence to be corrected is "हूअ सुठ आहे", where "सुठ" is incorrect, the process involves masking the incorrect word (e.g., "हूअ [MASK] आहे") and feeding it into the model. The model then predicts the correct word to update the sentence.

Table 2: Examples of Sindhi Devanagari Text Lines

S.No.	Sindhi Devanagari	Hindi	English
(i)	हूअ सुठो आहे	वह अच्छी है	She is good.
(ii)	हूअ सुंदरु आहे	वह सुंदर है	She is beautiful.
(iii)	हूअ हुशियार आहे	वह होशियार है	She is intelligent.
(iv)	हूअ मासूमु आहे	वह निर्दोष है	She is innocent.
(v)	हूअ आलसी आहे	वह आलसी है	She is lazy.

Consider the scenario where the error correction model predicts words like "हुशियार," "मासूमु," "सुठो," "सुंदरु," and "आलसी" from highest to lowest score as shown in Table 3. If the original token is replaced with the word having the highest score, the incorrect sentence "हूअ सुठ आहे" would incorrectly become "हूअ हुशियार आहे". This means that, rather than correcting the incorrect word, the word has been entirely replaced. While the new word is contextually appropriate, the original meaning of the sentence is lost.

Table 3: Predictions and score for "हूअ [MASK] आहे"

score	token	token_str
0.1028161358833313	507	हुशियार
0.0811197263230324	636	मासूमु
0.07516263113088608	3231	सुठो
0.04072516828775406	2290	सुंदरु
0.00279793947935104	2478	आलसी

To prevent such errors, the Levenshtein distance is applied to the top five candidates. This approach measures the similarity between the original and predicted tokens, selecting

only those with a similarity score greater than or equal to 0.60 for *Sindhi_Testdataset_A*.

For *Sindhi_Testdataset_B*, which belongs to a different domain than the training data, the Levenshtein distance has been reduced to 0.50. Among these, the word with the highest similarity score is then chosen as the most appropriate replacement. In cases where the error correction system does not suggest any suitable alternatives, the original word remains unchanged. This whole process is explained in Figure 2. In this figure, the incorrect sentence "इहा असां पहिंज अखियुनि सां डि॒ठी आहे ।" from OCR is corrected using the proposed 'Sindhi_Dev_Error_Correction_Model'. The Ground Truth for this sentence is "इहा असां पंहिंजे अखियुनि सां डि॒ठी आहे ।". Examples illustrating these scenarios are discussed in section 6.

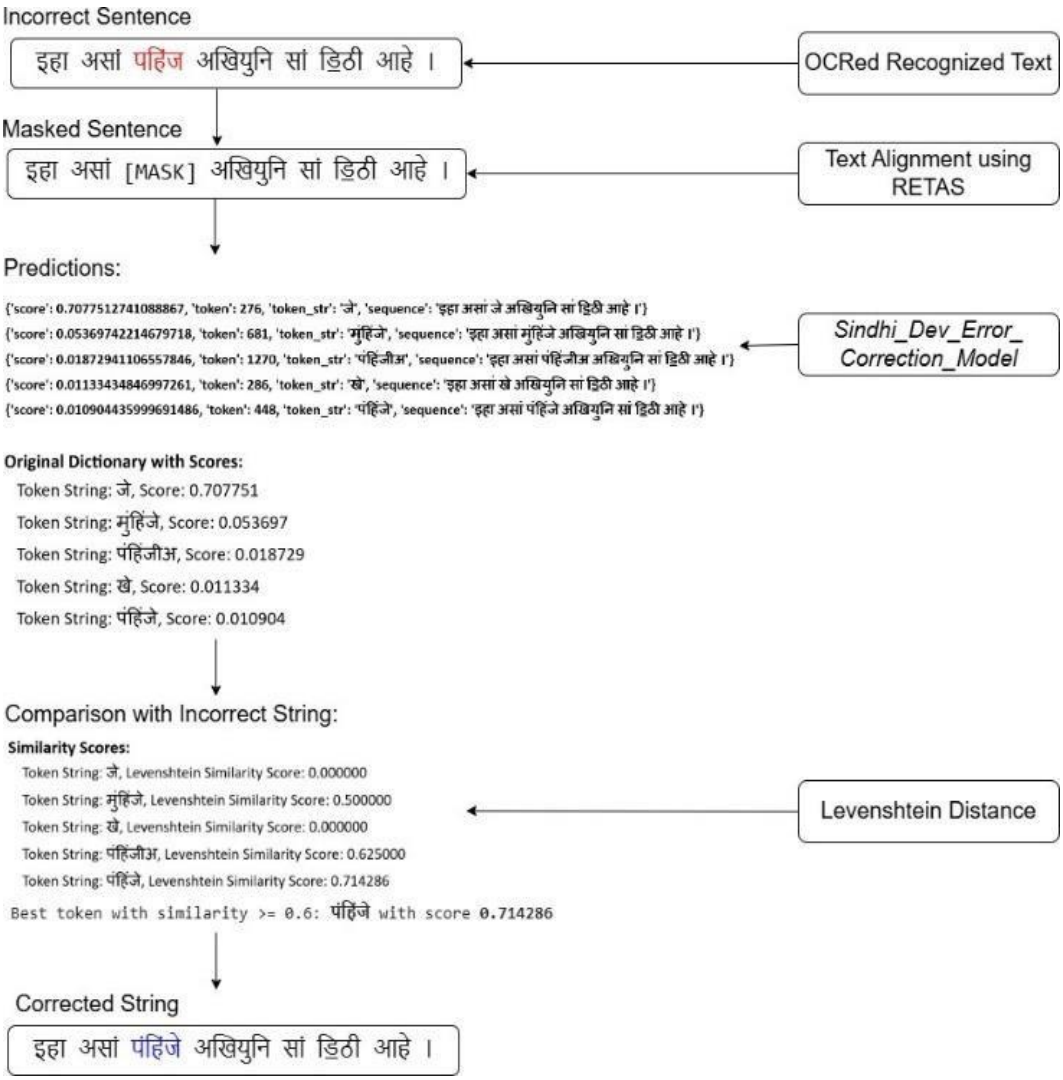


Figure 2: Overall Research Methodology Process

6. Experimental results and Analysis

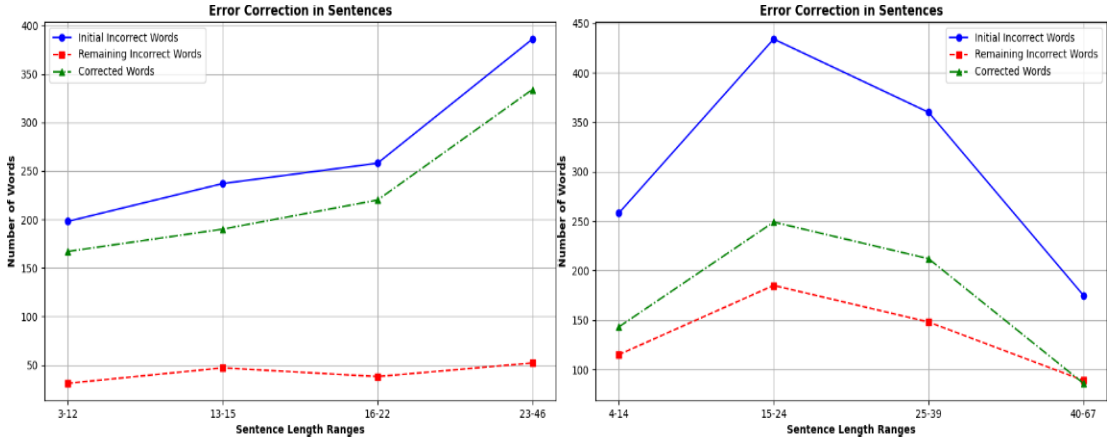
6.1 Performance of trained BERT Model on *Sindhi_Testdataset_A* and *Sindhi_Testdataset_B*

The proposed error correction approach was evaluated using *Sindhi_Testdataset_A* and *Sindhi_Testdataset_B*, as outlined in Section 4. The evaluation metric applied was Word Error Rate, Character Error Rate and Accuracy.

Word Error Rate: It is calculated by dividing the total number of incorrect words by the total word count in the ground truth text, and is represented as:

$$WER(\%) = \frac{Number\ of\ incorrect\ words}{Total\ number\ of\ words} \times 100$$

The line chart in Figure 3 visualizes error correction in sentences of varying lengths in *Sindhi_Testdataset_A* and *Sindhi_Testdataset_B* respectively. It compares the number of initial incorrect words with the remaining incorrect words after post-processing and calculates the corrected words.



(a) *Sindhi_Testdataset_A* (b) *Sindhi_Testdataset_B*

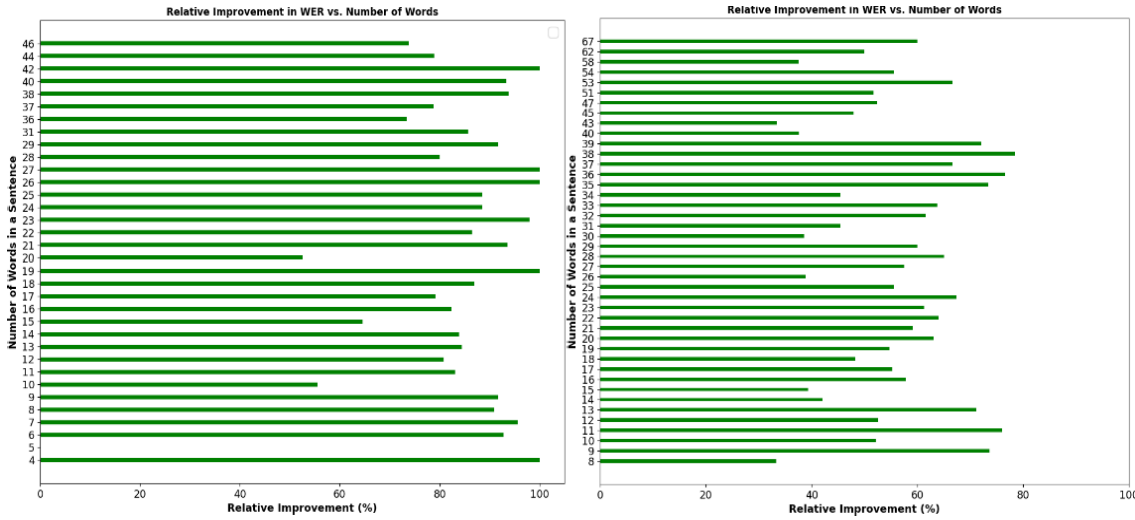
Figure 3: Comparison of Initial and Remaining Incorrect Words After Post-Processing

The relative improvement in Word Error Rate (WER) across different sentence lengths for our datasets, *Sindhi_Testdataset_A* and *Sindhi_Testdataset_B*, is shown in Figure 4. Relative improvement refers to the percentage reduction or gain in a value compared to its initial value, highlighting the extent of change or progress from the starting point.

Character Error Rate: Character Error Rate (CER) is calculated using the Edit Distance (Levenshtein distance). This metric is defined as the ratio of Insertion (I), Substitution (S), and Deletion (D) errors to the total number of characters, and is given by:

$$CER(\%) = \frac{I + S + D}{Total\_Characters} \times 100$$

In this context, S indicates the number of substitutions, D represents the number of deletions, and I refer to the instances of insertions. The Total Characters corresponds to the annotation size or the total count of characters in the reference text. The Edit Distance is used to measure the difference between two strings (e.g., words) by determining the minimum number of operations required to convert one string into the other.



(a) *Sindhi_Testdataset_A* (b) *Sindhi_Testdataset_B*

Figure 4: Relative Improvement in Word Error Rate (WER) for Different Sentence Lengths

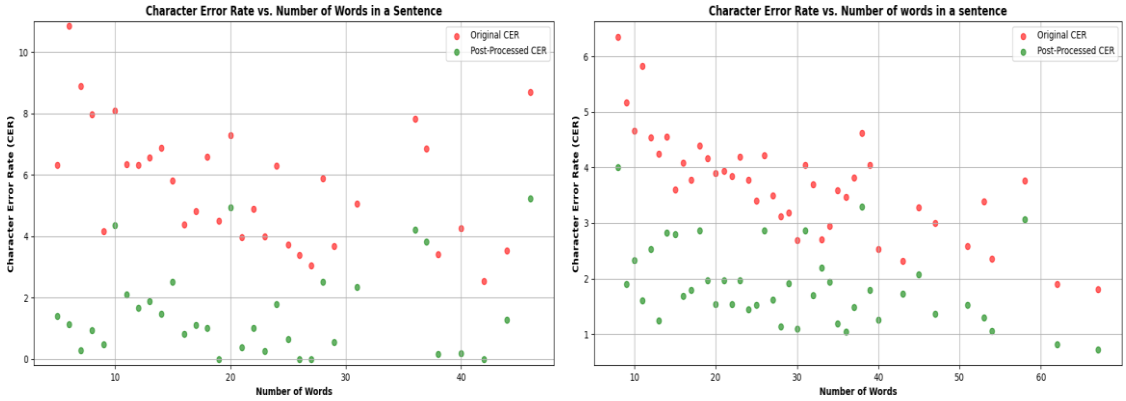
The Character Error Rate in Figure 5 visualizes the relationship between sentence length and Character Error Rate (CER) using a scatter plot. It compares the original CER with the post-processed CER for different document lengths. These results demonstrate a clear improvement in accuracy for both testing datasets. In *Sindhi_Testdataset_A*, a higher concentration of red and green dots indicates less improvement, however, a lower concentration suggests that the CER has improved to a satisfactory level.

Accuracy: Accuracy measures the percentage of correctly recognized characters by the post-processing system relative to the total characters in the dataset and is expressed as:

$$Accuracy\ (\%) = \frac{Total\ no\ of\ correct\ characters}{Total\ no\ of\ characters} \times 100$$

Sindhi Devanagari OCR achieved accuracy rates of 94.45% on *Sindhi_Testdataset_A* and

96.19% on *Sindhi_Testdataset_B*. After applying the 'Sindhi_Dev_Error_Correction_Model' to *Sindhi_Testdataset_A*, the accuracy improved from 94.45% to 98.46%, representing a 4.01% increase. Similarly, applying error correction to *Sindhi_Testdataset_B* raised the accuracy from 96.19% to 98.09%, a 1.90% improvement, as detailed in Table 4.



(a) *Sindhi_Testdataset_A* (b) *Sindhi_Testdataset_B*

Figure 5: Character Error Rate (CER) for Sentences of Varying Lengths: Original vs. Post-Processed

Table 4: Performance of 'Sindhi_Dev_Error_Correction_Model' on Test Datasets

Dataset	Accuracy	
	Sindhi Devanagari OCR	Sindhi_Dev_Error_ Correction_Model
Sindhi_Testdataset_A	94.45%	98.46%
Sindhi_Testdataset_B	96.19%	98.09%

These results demonstrate a clear improvement in accuracy for both testing datasets.

6.2 Error Analysis

A systematic analysis of all recognition errors in the output of our post-correction model is conducted to identify the sources of improvement over the OCR output and the factors affecting accuracy. Additionally, the types of errors introduced during the post-correction process are also investigated. In Figure 6, the examples of errors corrected through post-processing are provided.

While the performance of the trained model is promising, some errors are introduced by the model during the correction process which are presented in Figure 7. After post-processing with the MLM-BERT model, some incorrect words were replaced with completely different word from the dictionary. In *Sindhi_Testdataset_A*, 1.49% of the words, and in *Sindhi_Testdataset_B*, 11.34% of the words were replaced by the valid words as per dictionary but they are contextually incorrect, with some correct characters within the words being replaced as well.

The findings of this research demonstrate that the proposed model provides precise recommendations, as evidenced by the data in Table 5. For instance, the Sindhi Devanagari OCR mistakenly recognizes the word "हुओ" as "हुआ," and the 'Sindhi_Dev_Error_Correction_Model' accurately corrects it back to "हुओ" as shown in Table 6. In this, if there are more than one tokens having same similarity score greater than or equal to the threshold, then the token occurring first will be taken into consideration. Conversely, the incorrect word "हिस" is corrected to "होस" instead of "होसि" because the Levenshtein Distance between "हिस" and "होस" is smaller compared to other predictions as presented in Table 7.

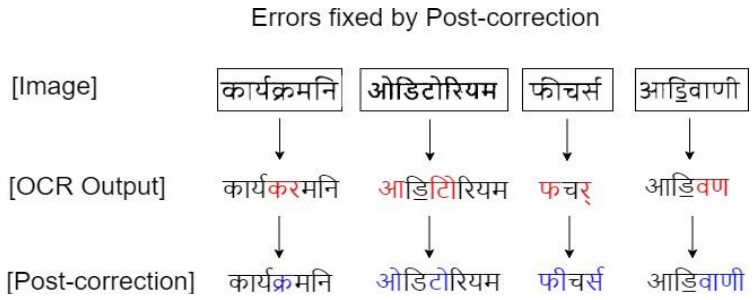


Figure 6: Summary of Errors Corrected Through Post-Processing

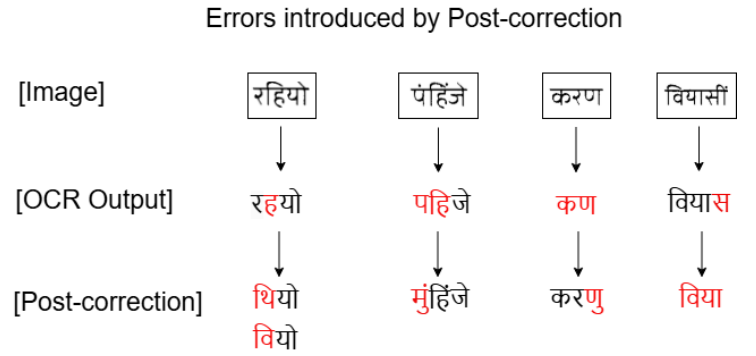


Figure 7: Analysis of Errors Introduced During Post-Processing

Furthermore, the trained model demonstrates the ability to make context-based corrections, as shown in Table 8. In Example 1, the word "नाक," though correct in isolation, is contextually incorrect, and the model successfully corrects it based on the surrounding context.

Similarly, in Example 2, OCR may introduce errors by adding unnecessary characters to a word, rendering it incorrect. The model leverages contextual information to rectify these errors as well. Sometimes, the error correction model struggles to provide accurate predictions for words that include special characters, such as the hyphen in "रक्षा-बंधन", "ड्डहंनि-पंद्रहनि", "टे-फरसला",etc. A hyphen is used to link words or parts of words. For instance, the OCR

Table 5: Precise Recommendations by the Proposed Model

Actual word	OCR Recognized output as Input	*Sindhi_Dev_ Error_Correction_Model* output
हुओ	हुआ	हुओ
हुआईं	हुईं	हुआईं
मायूस	मामूय	मायूस
आया	आुया	आया
करायो	कायो	करायो
कुझ	कुइ	कुझ

output is "छा-बंधन," instead of "रक्षा-बंधन" but the correction model fails to make the correct predictions. Instead, it might suggest words like "रक्षा," "कलाकारनि," "रेकार्डिंग," "फले," and "बंधन". This issue also extends to hyphenated dates, such as ”13-13-2003,” where the model is unable to generate the correct prediction.

Table 6: Accurate replacement of incorrect word with the correct prediction for the sentence: ” हू हिन्दीअ में ब कविताऊं लिखंदो हुआ । ”

Ground Truth	हू हिन्दीअ में बि कविताऊं लिखंदो हुओ ।	
OCR Recognized Sentence	हू हिन्दीअ में ब कविताऊं लिखंदो हुआ ।	
Masked Sentence	हू हिन्दीअ में बि कविताऊं लिखंदो [MASK] ।	
Predictions from *Sindhi_Dev_Error_ Correction_Model*	Token String	Score
	हो	0.997965
	हुओ	0.000824
	आहे	0.000440
	हुई	0.000220
	हुजे	0.000090
Similarity Score using Levenshtein Distance of Incorrect word with predictions	Token String	Levenshtein Similarity Score
	हो	0.333333
	हुओ	0.666667
	आहे	0.000000
	हुई	0.666667
	हुजे	0.400000
Corrected Sentence	हू हिन्दीअ में बि कविताऊं लिखंदो हुओ ।	

Table 7: Inaccurate replacement of incorrect word for the sentence: " नढपण खां ई नाटकनि जो शौंक हिस ।"

Ground Truth	नढपण खां ई नाटकनि जो शौंक होसि ।	
OCR Recognized Sentence	नढपण खां ई नाटकनि जो शौंक हिस ।	
Masked Sentence	नढपण खां ई नाटकनि जो शौंक [MASK] ।	
Predictions from *Sindhi_Dev_Error_Correction_Model*	Token String	Score
	होसि	0.992591
	हुओ	0.002652
	हो	0.000232
	होमि	0.000158
	होस	0.000104
Similarity Score using Levenshtein Distance of Incorrect word with predictions	Token String	Levenshtein Similarity Score
	होसि	0.500000
	हुओ	0.333333
	हो	0.333333
	होमि	0.250000
	होस	0.666667
Corrected Sentence	नढपण खां ई नाटकनि जो शौंक होस ।	

Additional examples are shown in Table 9, where the error correction model generates predictions but fails when the similarity score between the predicted word and the incorrect word is below the threshold.

After analyzing the errors, it has been observed that correction depends on several factors, including the degree of distortion in the incorrect word, the set threshold for the acceptance of predicted word, location of incorrect word in the sentence and the presence of consecutive errors in a sentence. When a word is severely distorted, the model may identify the correct word, but it is highly likely that the distance between the incorrect and predicted words will not meet the required threshold, leaving the word unchanged. Also, the model struggles when most of the words in the sentence are incorrectly recognized by the OCR, making it challenging for the model to accurately predict corrections based on the surrounding context.

7 Conclusion

This research paper addresses the challenge of automatic error correction for OCR outputs in Sindhi written in Devanagari script. A state-of-the-art error correction model using Masked Language Modeling (MLM) with BERT is proposed. The model selects the most suitable

Table 8: Output of 'Sindhi_Dev_Error_Correction_Model' indicating suitable output as per context.

Example 1	
Actual Text(Sindhi Devanagari)	बम्बईअ में हुन घणा नाटक पेश कया ।
Translation (Hindi)	उन्होंने बॉम्बे में कई नाटक किए।
Translation (English)	He performed many plays in Bombay.
OCR Recognized Text	बम्बईअ में हुन घणा नाक पेश कया ।
Sindhi_Dev_Error_ Correction_Model	बम्बईअ में हुन घणा नाटक पेश कया ।
Example 2	
Actual Text(Sindhi Devanagari)	सम्मेलन में जेको शामिल थिए सो नाठीअ जहिड़ो स्वागत तलबे ।
Translation (Hindi)	सम्मेलन में भाग लेने वालों का भी ऐसा ही स्वागत किया जाना चाहिए।
Translation (English)	Those who attend the conference deserve a similar welcome.
OCR Recognized Text	सम्मेलन में जेको शाल थिए सो नाठीअ अजहिड़ो स्वागत तलबे ।
Sindhi_Dev_Error_ Correction_Model	सम्मेलन में जेको शामिल थिए सो नाठीअ जहिड़ो स्वागत तलबे ।
Example 3	
Actual Text(Sindhi Devanagari)	हालतूं निहायत मायूस कन्दड़ हुयूं ।
Translation (Hindi)	परिस्थितियाँ बहुत निराशाजनक हैं।
Translation (English)	Conditions are very frustrating.
OCR Recognized Text	हूं निहायत मायूस कन्दड़ हुयूं ।
Sindhi_Dev_Error_ Correction_Model	हालतूं निहायत मायूस कन्दड़ हुयूं ।

Table 9: Examples where it fails due to low similarity score than the set threshold.

Ground Truth	OCR Recognized Output	Ground Truth	OCR Recognized Output
नंढिपिण	नढपण	शायल	शल
कामेटीअ	कामअ	घनशाम	ख़शनानम
आज़ाद	आद	नम़्मानिगार	जन्मनगिगार

word among the top five candidates based on their assigned probabilities. Next, a similarity score is determined using the Levenshtein Distance, and the incorrect word is replaced with the new word if the similarity score is equal or above the threshold. The model provides context-sensitive suggestions, a novel approach for this language. This automatic error correction model achieved a 4.01% accuracy improvement on the dataset from the same domain as the training set, and a 1.90% increase on the dataset from a completely different domain. Lowering the threshold value for *Sindhi_Testdataset_B* improves accuracy to a

certain degree but also leads to contextually incorrect word replacements in some instances.

It was observed that the MLM BERT model sometimes fails to provide appropriate suggestions, particularly for numbers or words containing hyphens (-). Additionally, the model struggles to offer corrections when most of the words in a sentence are incorrect. Future work will focus on increasing the dataset for training so that more domains can be covered, the accuracy enhancement of the model through ensemble approaches and exploring more effective language models for Sindhi Devanagari OCR error correction.

Declarations

Ethical Approval: This declaration is not applicable. Funding: No funding was received to assist with the preparation of this manuscript.

Financial Interests: The authors, Arvind Kaur and Gurpreet Singh Lehal declare that there are no financial interests for this research work.

Conflicts of Interest: The authors declare that they have no conflicts of Interest to report regarding the present study.

Data Availability Statement

The Authors have created their own dataset for the experimental study.

References

1. Vikas O. 2005. Issues in Representation of Indic Scripts in Unicode. L2/05-063). http://www.unicode.org.
2. Vinitha V S, Mathew M and Jawahar C. 2017. An Empirical study of Effectiveness of Post-processing in Indic Scripts. IAPR International Conference on Document Analysis and Recognition (ICDAR). 7: 32–36
3. Das D, Philip J, Mathew M and Jawahar C. 2019. A Cost Efficient Approach to Correct OCR errors in Large Document Collections. International Conference on Document Analysis and Recognition (ICDAR). 655–662
4. Saluja R, Adiga D, Chaudhuri P, Ramakrishnan G and Carman M. 2017. Error Detection and Corrections in Indic OCR using LSTMs. IAPR International Conference on Document Analysis and Recognition (ICDAR). 1: 17–22
5. Srigiri S and Saha K S. 2020. Spelling Correction of OCR-generated Hindi Text using Word Embedding and Levenshtein Distance. International Conference on Nanoelectronics, Circuits and Communication Systems: Proceeding of NCCS, Springer. 415–424
6. Saluja R, Punjabi M, Carman M, Ramakrishnan G and Chaudhuri P. 2019. Sub-word Embeddings for OCR Corrections in Highly Fusional Indic Languages. IEEE International Conference on Document Analysis and Recognition (ICDAR). 160–165
7. Virk M S, Dannélls D and Muhammad S A. 2021. A Novel Machine Learning Based Approach for Post-ocr Error Detection. Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP). 1463–1470
8. Nguyen H T T, Jatowt A, Coustaty M and Doucet A. 2021. Survey of post-OCR Processing Approaches. ACM Computing Surveys (CSUR). 54: 1–37
9. Bansal V and Sinha K M. 2001. A Complete OCR for Printed Hindi Text in Devanagari Script.

Proceedings of Sixth International Conference on Document Analysis and Recognition. 800–804

10. Pal U, Kundu KP and Chaudhuri BB. 2000. OCR Error Correction of an Inflectional Indian Language using Morphological Parsing. Journal of Information Science and Engineering 16: 903–922
11. Lehal S G, Singh C and Lehal R. 2001. A Shape Based Post Processor for Gurmukhi OCR. Proceedings of Sixth International Conference on Document Analysis and Recognition. 1105–1109
12. Rasagna V, Kumar A, Jawahar V C and Manmatha R. 2009. Robust Recognition of Documents by Fusing Results of Word Clusters. 10th International Conference on Document Analysis and Recognition. 566–570
13. Mohan K, Jawahar V C. 2010. A Post-processing Scheme for Malayalam using Statistical Sub-character Language Models. Proceedings of the 9th IAPR International Workshop on Document Analysis Systems. 493–500
14. Bojanowski P, Grave E, Joulin A and Mikolov T. 2017. Enriching Word Vectors with Subword Information. Transactions of the association for computational linguistics. 5: 135–146
15. Nguyen D Q, Le A D, Phan M N and Zelinka I. 2021. OCR Error Correction using Correction Patterns and Self-organizing Migrating Algorithm. Pattern Analysis and Applications, Springer. 24: 701–721
16. Chen S and Liao H. 2022. Bert-log: Anomaly Detection for System Logs Based on Pre-trained Language Model. Applied Artificial Intelligence. 36(1)
17. Saleh H, Alhothali A and Moria K. 2023. Detection of Hate Speech using Bert and Hate Speech Word Embedding with Deep Model. Applied Artificial Intelligence. 37(1)
18. Ni P and Wang Q. 2022. Internet and Telecommunication Fraud Prevention Analysis based on Deep Learning. Applied Artificial Intelligence. 36(1)
19. Xie G, Liu N, Hu X and Shen Y. 2023. Toward Prompt-enhanced Sentiment Analysis with Mutual Describable Information Between Aspects. Applied Artificial Intelligence. 37(1)
20. Riyadh M and Shafiq O M. 2022. GAN-BElectra: Enhanced Multi-class Sentiment Analysis with Limited Labeled Data. Applied Artificial Intelligence. 36(1)
21. Kundaikar T, Fadte S, Karmali R and Pawar D J. 2024. Automatic Hindi OCR error correction using MLM-BERT. International Information and Engineering Technology Association (IIETA). 619–626
22. Suseela J V and Uma V. 2014. Unicode Applications in the Digital Libraries of India. Current Practices in Academic Librarianship. 1:40
23. Yalniz Z I and Manmatha R. 2011. A Fast Alignment Scheme for Automatic OCR Evaluation of Books. International Conference on Document Analysis and Recognition, IEEE. 754–758
24. Devlin J, Chang WM, Lee K and Toutanova K. 2018. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT. 4171-4186
25. Kokonos L K. 2022. Creation of A New BERT Model Using Greek Legal Data. Thesis: National and Kapodistrian University of Athens. 1–41