International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 29, No. 2, April, 2024, pp. 619-626
Journal homepage: http://iieta.org/journals/isi

Automatic Hindi OCR Error Correction Using MLM-BERT

Teja Kundaikar*, Swapnil Fadte, Ramdas Karmali, Jyoti D. Pawar

Discipline of Computer Science & Technology, Goa Business School, Goa University, Goa 403206, Taleigao Plateau, India

Corresponding Author Email: teja.kundaikar@unigoa.ac.in

https://doi.org/10.18280/isi.290223

Received: 6 September 2023
Revised: 10 January 2024
Accepted: 22 January 2024
Available online: 25 April 2024

Keywords:
Hindi, devanagari, MLM BERT, OCR, error correction, natural language processing, deep learning, post-processing

ABSTRACT

Optical Character Recognition (OCR) systems find it challenging to generate accurate text for highly inflectional Indic languages such as Hindi. Inflectional languages possess an extensive vocabulary. Words in these languages can assume different forms based on factors like gender, meaning, or other contextual cues. To enhance the accuracy of OCR and correct the errors resulting from the inflectional nature of language, it is crucial to perform post-processing on output of the OCR. This work focuses on correcting errors in the OCR output specifically for the Hindi language. To overcome existing challenges, an error correction model has been proposed in this work that uses the Masked-Language Modeling with BERT (MLM-BERT). It utilizes the context to provide accurate word suggestions for the incorrect word or masked word. The proposed model has been tested using the Hindi OCR test dataset from IIITH. It achieved an improvement of 3.58% word accuracy over the baseline OCR word accuracy, which demonstrates its effectiveness in enhancing the accuracy of the OCR output text.

1. INTRODUCTION

Hindi, being a major regional language in India with over 500 million speakers worldwide, requires an accurate Optical Character Recognition (OCR) solution to address automatic document analysis problems. OCR refers to the identification of text from digitally scanned documents. Preserving the meaning of the text and the content can effectively be utilized for natural language applications. In the context of Indic languages such as Hindi, the accuracy of OCR output is often low and requires additional post-processing to refine it further. Hindi poses several challenges for OCR systems. These challenges include the complexity of the scripts (complex syllables formed by various combinations of vowels, consonants and conjunct consonants), the highly inflectional nature of the language [1], and the errors that can occur within the context of a sentence. In some cases, an incorrectly recognized word may still be valid in the same language, which can further complicate the task of OCR. Moreover, due to the poor availability of language resources, having good accuracy can make it difficult to train the systems.

Currently, available OCR for Hindi are Tesseract OCR [2], Indsenz OCR which is a commercial product [3], eAksharaya OCR [4] and Multilingual OCR for Indian languages [5]. Among these most reasonable OCRs are Tesseract OCR and Multilingual OCR for Indian languages. They showed around 89.29% and 91.6% word accuracy on the available Hindi OCR test dataset from IIITH, respectively [5-7]. The word accuracy achieved by them needs substantial improvement, which can be achieved through post-processing of the OCR output.

Recent post-processing approaches that have been utilized to improve the accuracy of Hindi OCR are dictionary-based [8], statistical language models [8, 9], Syllable and confusion matrix based [7], Deep learning LSTM [10], Word Embedding & Levenshtein distance [11] and sub-word embedding [12]. However, these approaches do not address contextual errors. To address this limitation, we propose a context-sensitive automatic error correction approach to improve the accuracy of Hindi OCR output. In this work, the goal is to correct errors found in scanned documents generated via Tesseract Hindi OCR system, specifically focusing on providing suggestions for incorrect words based on contextual information, as Tesseract is a freely available OCR providing reasonably good accuracy.

The paper is organized as follows: Section 1 briefly introduces challenges of Hindi OCR, emphasizes the significance of post-processing and briefly introduces the proposed error correction approach. Section 2 discusses the relevant existing work on OCR error correction approaches. The motivation behind the current work is outlined in section 3. Section 4 presents the methodology for developing an automatic error correction model for Hindi OCR. The experimental results and error analysis are presented in section 5. Lastly, the conclusion and future scope are mentioned in section 6.

2. RELATED WORK

Error correction is a necessary step to improve the accuracy of the OCR output text. This section discusses the various OCR error correction approaches used for post-processing the OCR output for Indian languages.

Post-processing techniques have been extensively used in the past to enhance OCR accuracy. Various OCR post-processing techniques are explored by Nguyen et al. [13]. 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.

Dictionary-based approach on the Hindi OCR recognized document showed a character accuracy of 93% [14] The problem with the dictionary-based approach is that it assumes the word suggested as true if it is present in the dictionary and also it cannot correct the contextual errors.

Syntactic and semantic-based approaches for error correction can be found here. A Morphological analyzer was utilized for Bangla (Indian script) to find the root words and provide correct suggestions [15]. The corrections made here are based on a fast dictionary access method and suggest the correct word in 82.44% of cases.

The shape-based post-processing technique was applied to Gurmukhi (Indian script) and achieved a 3% improvement, attending to 97.34% character accuracy [16]. Here, consonants are categorized into groups based on shape similarity, each assigned a unique number. The input word is encoded accordingly. If the code is found in the dictionary, a matching operation is executed between the input word and words stored under that specific code. In cases without a match, structurally similar words to the input are suggested as alternatives. This approach finds difficulty in providing suggestions to similar characters.

Locality Sensitive Hashing (LSH) was used to create word clusters to improve the Telugu language OCR accuracy [17], resulting in 79.12% word accuracy. Here the OCR outputs of word images within a cluster are compared, and improvements are achieved through methods such as Character Majority Voting or Dynamic Time Warping Technique. This technique fails to provide suggestions for unique words (words appearing only once in a cluster).

The Weighted Syllable-based Spell Correction Approach (WSSC) was applied to Hindi OCR-recognized text, resulting in 88.22% word accuracy [7]. In this method, a syllable confusion matrix was generated, taking into account the frequency of each syllable substitution. Word was looked up in the dictionary to check whether it is correct or incorrect. If the word is incorrect, then based on the confusion matrix, appropriate syllables are substituted for all syllables in the words. Forming many words and assigning 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. It will not suggest contextual suggestions. All these OCR post-processors implement post-processing at the Unicode (or character) level.

A graph-based, sub-character level language model was applied to rectify OCR word errors in Malayalam [18]. In this approach, edges represent the language information, and nodes represent visual similarities. The optimal path from source to destination provides suggested words. Through this approach word accuracy of 95.0% is achieved [18]. This approach fails to provide suggestions for rare words, proper nouns and foreign words.

Statistical language models (SLM) and Dictionary-based methods have been used in error correction for Indian languages such as Hindi, Punjabi (Gurmukhi script), Telugu, and Malayalam [9]. This approach reported Improvements in errors in ranges between 38% to 66% for the above languages. The limitation of this work is that a dictionary and character n-gram alone cannot effectively suggest the correct word due to the inflectional nature of Indian languages.

Sub-word embeddings have also been applied for OCR error correction in Hindi, Sanskrit, Kannada, and Malayalam, resulting in 90.42% word accuracy for Hindi [12]. Further intersection of error rectification and cluster analysis was observed in the clustering of similar words within documents for error correction, particularly in Hindi and English [8]. This approach lacks contextual suggestion.

Spelling correction in OCR-generated Hindi text, utilizing methods like Word Embedding and Levenshtein Distance [11]. This uses the Continuous Bag-of-Words (CBOW) model. It does not consider the order of words in a given context. Thus, it will struggle to provide contextual suggestions for incorrect words.

Recently, deep learning techniques were used for OCR error detection and correction spanning four languages—Hindi, Malayalam, Sanskrit, and Kannada—specifically an LSTM approach, particularly relevant when languages are written in distinct scripts [10]. It showed 92% of word accuracy for Hindi. LSTMs can capture dependencies over longer sequences than simple RNNs, but they still have a limited context window.

The above literature review indicates that post-processing of OCR output for Indian languages is challenging due to context-dependent errors and the inherent nature of highly inflectional languages. The abundance of words due to the inflectional nature of the Indian language also poses a challenge in OCR error correction.

3. MOTIVATION

Literature reviews have shown that there is still scope for improving the OCR output of Indian languages. The review of the existing literature has indicated that post-processing in OCR output is crucial in enhancing the performance of Hindi OCR. However, the most commonly used approaches do not address context-sensitive error correction, as indicated by the three examples shown in Table 1. The red color indicates an incorrect word. Example 1 where the incorrect word is कां. Here, the existing bigram model can show suggestions as का, के, की, etc. Likewise, in example 2, रनप is an incorrect word. Existing models can suggest a word like बाद, साथ, लिए etc. Similarly, in example 3 the incorrect word is एफटीआहँआई, which is an English foreign word. For such cases, we need an approach that can suggest a word based on context.

This can be achieved using state-of-the-art techniques such as Masked Language Model (MLM) with Bidirectional Encoder Representations from Transformers (BERT), which have not yet been explored for OCR error correction in Indian languages. The MLM BERT model has been utilized for tasks such as detection [19-21] and correction [22], demonstrating its ability to simplify the model’s architecture and provide contextual suggestions [23]. MLM BERT masks words containing errors and offers suggestions for the masked word, based on available contextual information. To build the MLM BERT for Hindi, it is necessary to have a BERT tokenizer and MLM BERT model for Hindi. In this paper, an OCR error correction approach using an advanced MLM with BERT to suggest contextually aware correction is discussed in section 4.

Table 1. Some examples of OCR recognized errors that require contextual suggestion

Example 1	
Actual text (Hindi)	उत्पादों को पेश कर रहे हैं।
Transliteration (English)	*UTPADOM KO PESa CaRa RaHE HAiM.*
Translation (English)	Presenting products.
OCR recognized text	उत्पादों कां पेश कर रहे हैं
Example 2	
Actual text (Hindi)	वायसराय के अंगरक्षक के रूप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।
Transliteration (English)	*VAYaSaRAYa KE aMGaRaKSaKa KE RuPa mEM GaThita YaHa BhARatIYa SENA KA SaBaSE PuRANA REJImEMTa HAi.*
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 रनप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।.
Example 3	
Actual text (Hindi)	एफटीआईआई की वर्तमान छात्रा
Transliteration (English)	*EFaTIAI KI VaRTaMANa ChATRA*
Translation (English)	Current student of FTII
OCR recognized text	एफटीआहँआई की वर्तमान छात्रा

4. METHODOLOGY

This section discusses two main parts of the experimental setup. The first part describes how the dataset was created for testing the error correction model. The second part provides details of the experimental setup.

4.1 Test dataset creation

The procedure to create datasets to evaluate the error correction model is discussed here. Two sets of test data were utilized to evaluate the performance of the proposed error correction model. The first dataset is the Hindi OCR dataset obtained from IIITH [5]. It comprises a total of 100 pages presented in image format, along with their respective text files. For the sake of convenience, we have named this dataset as *Hindi-IIITH-dataset.* This dataset showed 91.6% word accuracy on OCR for Indian languages [5] and 89.29% on Tesseract Hindi OCR [7]. The second dataset is created from a part of Hindi text within the same domain, which has been utilized for training in this work. The total number of words available in the *Hindi-IIITH-dataset* is 29884. The characteristics of this dataset is presented in Table 2. It shows the distribution of words based on the type of syllable present in the word. The eight categories of words found in the dataset are presented below -

Words without modifier: Words with Devanagari numerals or vowels or consonants with no modifiers. e.g. मन (*maNa*), हलचल (*HaLaCaLa*), 2023.

Words with Matra: Words with *Matra* (Matra is a Dependent vowel sign. ◌े([e]), ◌ै([ai]), ◌ो ([o]), and ◌ौ ([au]) in Devanagari [24]) e.g. वैर (*VAiRa*).

Words with Eekar: Words with modifier *Eekar* (Eekar is a Dependent vowel sign. ि◌ ([i]) and ◌ी ([e:]) in Devanagari [24]) e.g. सीप (*SIPa*).

Words with Ookar: Words with modifier *Ookar* (Ookar is a Dependent vowel sign. ◌ु([u]) and ◌ू([u:]) in Devanagari [24]) e.g. अनुभव (*aNuBhaVa*).

Words with Conjunct consonant: words, which have *Conjunct consonant* (Conjunct consonant have more than two consonant letters [25]) e.g. पल्लव (*PaLLaVa*).

Words with Anusvara: Words having Anusvara (Devanagari sign ◌ं (Anusuvar. Bindi) [24]) e.g. पंप (*PaMPa*).

Words with Candra: Words with Candra (Devanagari sign ◌ँand letter "Candra A" [24].) e.g. चाँद (*CaMDa*).

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 (*manIKA*), नितारा (NITARA).

Table 2. Characteristics of data used to create dataset to test error correction model

Category/Dataset	*Hindi_test_dataset*	*Hindi-IIITH-dataset* [5]
Without modifiers	5274	9930
Matras	7140	7849
Eekar	4872	4723
Ookars	792	1340
Conjunct consonant	1869	1083
Anusvara	571	248
Candra	65	298
Words with multiple modifiers	9301	5868
Total	29884	31339

The process that is used to create the Hindi test dataset (*Hin_test_dataset*) to test proposed error correction model is discussed below:

(1) The part of Hindi text sourced from [26] containing 29884 words is saved in the file named merged_text. The merged_text document was divided into total *N_text* files, where each file consists of 22 lines. This splitting was done to ensure that the text could be accommodated on a standard A4-sized sheet of paper.

(2) Additionally, the *N_text* files underwent a conversion process to generate *N_Img* files in the ".png" format. The conversion involved utilizing the Nakula font style which is of size 12pt and a resolution of 150 DPI "X" and "Y" dimensions [27, 28].

(3) The data generated in steps 1 and 2 was named as *Hin_test_dataset* having *N_text* and *N_Img* Hindi files.

(4) Next the *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. 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) [29]. It does alignment at both the word and character levels. We utilized word-level alignment. The aligned words, i.e., the ground truth word and OCRed word, are checked. If these two words are equal, then OCRed word is assigned label 0. Otherwise, the word gets labeled 1.

The *Hin_test_dataset* has been created for the purpose of testing the proposed error correction approach on data within the same domain on which it is trained. The proposed error correction approach is discussed in section 4.2.

4.2 Experimental setup for error correction approach

BERT originally developed for two main tasks: next sentence prediction and MLM to predict the masked word in the sentence. The word to be predicted is masked with a token [MASK] and the model generates contextual suggestions for the masked word [30].

In this proposed error correction approach, we initially mask the incorrect word, and further, the proposed error correction approach suggests the contextually appropriate word for the masked word. Figure 1 describes the proposed system, which consists of three modules: I) Training BERT tokenizer, II) Training MLM BERT model, and III) Error Correction using MLM BERT.

A detailed explanation is provided for each phase involved in the process:

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. 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 a 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, we can convert subwords to unique IDs using the BERT tokenizer. The BERT tokenizer for Hindi has been created using *BertWordPeiceTokenizer* and the respective data which is shown in Table 3.

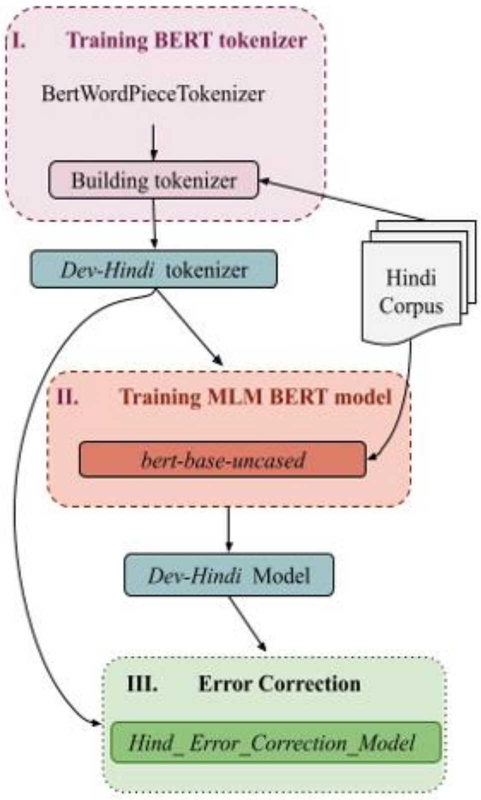

Figure 1. Proposed hindi error correction system

Table 3. Data used for Training *Dev-Hindi* tokenizer and *Hindi_Error_Correction_Model*

Language	Total Words	Source of Data
Hindi	8215774	Hindi dataset [24]

II. Training MLM BERT model: Here the BertForMaskedLM architecture is employed specifically utilizing the BERT Model known as *'bert-base uncased'*. Here we have chosen '*bert-base uncased' as* Devanagari is an uncased script. This particular 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. Despite utilizing this model, it was challenging to achieve satisfactory accuracy in correcting OCR errors in Hindi text. To support Hindi text, the BERT *Dev-hindi* 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 employed, which is the same as the one used during Phase I for constructing the tokenizer. The training parameters were configured as follows: Gaussian Error Linear Unit (GELU) as 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 539134 and the total number of optimization steps was 50500. The GELU was chosen as an activation function as it is a high-performing neural network activation function [31]. The choice of dropout rate and epoch configuration was influenced by a study demonstrating good performance results in the context of BERT [32]. After training the training loss was 1.54.

Table 4. Example of *Hindi_Error_Correction_Model* output Hindi language

Actual Word	OCR Recognized Word as Input	Top 6 Suggestion by *Hindi_Error_Correction_Model*	Probability	Proposed Model Output *
नई दिल्ली	राई 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*

III. Error Correction using MLM BERT: In this module, a lookup dictionary is utilized to identify incorrect words and the correction model suggests the appropriate word for every incorrect word. All the incorrect words were masked using [MASK] token. The word correction was achieved by utilizing the *"Dev-Hindi"* model and its corresponding tokenizer i.e. *"Dev-Hindi"*. Initially, the input OCR text having masked token is tokenized using the *"Dev-Hindi"* tokenizer, resulting in token IDs, and the *"Dev-Hindi"* 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 *"Dev-Hindi"* tokenizer. 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 the system as the most suitable word. An example is shown in Table 4. The word selected by the 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. Examples for each of these cases are discussed in section 5. 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.

5. EXPERIMENTAL RESULTS AND ANALYSIS

5.1 Performance of *Hindi_Error_Correction_Model* on *Hin_test_dataset* and *Hindi-IIITH-dataset*

The proposed error correction approach has been evaluated with *Hin_test_dataset* and *Hindi-IIITH-dataset* [5], the characteristics of which are detailed in Table 2. The ISRI analytic tool was used to calculate the word accuracy during the evaluation process [33]. The following evaluation metrics are used:

(i) Accuracy (word accuracy): 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.*

Table 5. Performance of *Hin_test_dataset and Hindi-IIITH-dataset*

Dataset	**Word Accuracy**	
	Tesseract OCR (Hindi)	**Hindi_Error_ Correction_Model**
Hin_test_dataset	87.23%	89.65%
Hindi-IIITH-dataset [5]	89.29%	95.18%

Table 6. Category-wise word error rate (WER) on *Hin_test_dataset* and *Hindi-IIITH-dataset*

	Hin_test_dataset		***Hindi-IIITH-dataset [5]***	
	Tesseract Hindi OCR Error %	***Hindi_Error_Correction_Model* Error %**	**Tesseract Hindi OCR Error %**	***Hindi_Error_Correction_Model* Error %**
Without modifiers	1.38	0.45	3.5	0.67
Matras	2.11	1.35	2.14	1.02
Eekar	0.96	0.82	0.92	0.75
Ookars	0.13	0.12	0.14	0.12
Conjunct consonant	0.99	0.87	0.98	0.5
Anusvara	0.68	0.68	0.65	0.65
Candra	0.05	0.05	0.03	0.03
Words with multiple modifiers	6.47	6.01	2.35	1.08
Total WER	12.77	10.35	10.71	4.82

It was observed that Tesseract Hindi OCR word accuracy on *Hin_test_dataset* and *Hindi-IIITH-dataset* were 87.23% and 89.29%, respectively. On performing error correction using *Hindi_Error_Correction_Model* on the dataset *Hin_test_dataset* the word accuracy increased from 87.23% to 89.65%. Additionally, when conducting error correction on the *Hindi-IIITH-dataset*, using the *Hindi_Error_Correction_Model*, it was observed that word accuracy increased from 89.29% and 95.18%. as shown in Table 5. 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* [5] has been noticed. It 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 [5].

Table 6 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. Our 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.

5.2 Error analysis

The results of this research work indicate that the proposed model delivers accurate recommendations, as demonstrated by the data presented in Table 7. For example, the actual word को *(KO)* is recognized as कां *(KAM)* via tesseract OCR systems, whereas the *Hindi_Error_Correction_Model* corrects it to को *(KO)*.

Table 7. Performance of *Hin_test_dataset* and *Hindi-IIITH-dataset*

Actual word	**OCR recognised output**	***Hindi_Error_Correction_ Model* output**
को	कां	को
और	ओंर	और
इन	हन	इन
कहा	केहा	कहा
इजरायल	इज़रायल	इजरायल
बधाई	वाई	बधाई
दौरा	दोस	दौरा
साथसाथ	साथस्नाथ	साथसाथ
इस	हस	इस

Let us consider another example to illustrate the effectiveness of the *Hindi_Error_Correction_Model*: The actual word बधाई *(BaDhAI)* is recognized by OCR as वाई *(VAI),* while the correction provided by the suggested model is बधाई. Additionally, the *Hindi_Error_Correction_Model* exhibits contextual-based corrections, as evidenced by the information provided in Table 8 and Table 9. For example, the actual text 6 हजार *(6 HaJaRa)* was incorrectly recognized as ह हजार *(Ha HaJARa),* and the proposed model suggests 10 हजार *(HaJARa).* However, the suggested word is incorrect. It is interesting to note that the suggested word is number and not any other word close to ह *(Ha).*

The *Hindi_Error_Correction_Model* also provides incorrect suggestions including cases where incorrect suggestions are given based on the context, as illustrated in Table 10. However, the suggested word is very similar to the correct word or contextually appropriate. Here is an example of the observation, i.e., the actual word विचारों *(VICAROM)* and OCR recognized word as विचाारो *(VICARO)* and *Hindi_Error_Correction_Model* suggestion is विचारो *(VICARO)* which is a quite similar suggestion to the actual word. Another example of a similar suggestion is with actual word उद्घाटन *(UDGhATaNa)*, OCR-recognized word उद्धाटन *(UDGhaTaNa)* and proposed model suggestion is उदघाटन *(UDaGhATaNa)*.

Table 8. Output of *Hindi_Error_Correction_Mode*l indicating suitable output as per context

Actual text (Hindi)	उत्पादों को पेश कर रहे हैं।
Transliteration (English)	*UTPADOM KO PESa CaRa HAiM.*
Translation (English)	Presenting products.
OCR recognized text	उत्पादों कां पेश कर रहे हैं
Hindi_Error_Correction_Model 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)	*VAYaSaRAYa KE aMGaRaKSaKa KE RuPa mEM GaThita YaHa BhARatIYa SENA KA SaBaSE PuRANA REJImEMTa HAi.*
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 रनप में गठित यह भारतीय सेना का सबसे पुराना रेजीमेंट है।.
Hindi_Error_ Correction_Model 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)	6 *HaJARa NAuCaRIYOM CA SRUJaNa HOGA*
Translation (English)	6 thousand jobs will be created.
OCR recognized text	ह हजार नौकरियों का सृजन होगा ।
Hindi_Error_Correctio n_Model Output	10 हजार नौकरियां का सृजन होगा।

Table 9. Output of *Hindi_Error_Correction_Model* showing correct suggestion

Actual text (Hindi)	पोत परिवहन मंत्रालय
Transliteration (English)	*POTa PaRIVaHaNa maMTRALaYa*
Translation (English)	Ministry of Shipping
OCR recognized text	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
OCR recognized text	एफटीआहँआई की वर्तमान छात्रा
Hindi_Error_Correction _Model Output	एफटीआईआई की वर्तमान छात्रा

Furthermore, an instance that demonstrates contextual-based recommendations is when real words is इसे *(ISE)*, OCR recognised word is हरने *(HaRaNE)* and model suggested the word as महामहिम *(maHAmaHIma)*, where the system suggested word was suitable for the sentence as shown in Table 11.

Table 10. Output of *Hindi_Error_Correction_Model* having quite similar suggestion to actual word or appropriate based on context

Actual word	**OCR Recognised Output**	**Model Correction**
आयोजन	प्रायोजनाा	परियोजना
गांवों	मांओं	भारत
उद्घाटन	उद्धाटन	उदघाटन
विचारों	विचाारो	विचारो
करोड़	कसड	करोड

Table 11. Output of *Hindi_Error_Correction_Model* although incorrect suggestion

Actual text (Hindi)	27 जनवरी 1950 को इसे राष्ट्रपति का अंगरक्षक नाम दिया गया।
Transliteration (English)	*27 JaNaVaRI 1950 KO ISE RAsRaPaTI KA aMGaRaKsaKa NAma DIYA GaYA*
Translation (English)	On 27 January 1950, it was named the President's Bodyguard.
OCR recognized text	17 जनवरी 1950 को हरने राष्ट्रपति का अंगरक्षक नाम दिया गया।
Hindi_Error_Correction_Model Output	17 जनवरी 1950 को महामहिम राष्ट्रपति का अंगरक्षक नाम दिया गया।

And cases where suggestions were not offered for words are depicted in Table 12 and Table 13. For example, having actual word हिस्सा *(HISSA)* and OCR recognized word द्विरुसा *(DVIRUSA)*, the proposed model did not provide any suggestion for word द्विरुसा *(DVIRUSA)*, result in retaining same word. Thus, it is observed that *Hindi_Error_Correction_Model* fails for numbers and mostly words whose first character of word is incorrectly recognized. Additionally, when the majority of neighboring words were incorrect, the *Hindi_Error_Correction_Model* fails to provide appropriate suggestions, as shown in Table 14.

Table 12. Output of *Hindi_Error_Correction_Model* provided no suggestion

Actual text (Hindi)	2008 में
Transliteration (English)	*2008 mEM*
Translation (English)	In 2008
OCR recognized text	1008 में
Hindi_Error_Correction_Model Output	1008 में

Table 13. Examples of *Hindi_Error_Correction_Model* output providing no suggestion

Actual Word	**OCR Recognised Output**	**Model Correction**
हिस्सा	द्विरुसा	द्विरुसा
रस्मी	रख्बेमी	रख्बेमी
स्वागत	रत्रागत	रत्रागत
प्रणाली	प्याली	प्याली

Table 14. Output of *Hindi_Error_Correction_Model* with incorrect 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.
OCR recognized text	गणमानय व्यक्ति भी हस अबरार पर उपस्थित थे।
Hindi_Error_Correctio n _Model Output	गणमानय व्यक्ति भी शरी अबरार पर उपस्थित थे।

6. CONCLUSION AND FUTURE SCOPE

In this research paper, the problem of automatic error correction of OCR output of highly inflectional Hindi language is addressed. Error correction model utilizing state-of-art Masked Language Modeling (MLM) with BERT is proposed. The model selects the best word among the top 6 candidates based on assigned probabilities. The proposed automatic error correction model improved word accuracy by 3.58% over Tesseract OCR. There is more improvement in words without any modifiers. And negligible improvement for the words consisting of Ekar, Ookar, Conjunct consonant and words with multiple modifiers. Model outperforms its counterparts 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, such as numbers, the MLM BERT model fails to provide appropriate suggestions. The model does not provide any suggestions especially when there exist two or more incorrect consecutive words. Also if most of the neighboring words are incorrect with very low edit distance measure then the model is not able to provide proper suggestions. Moving forward, future work will prioritize improvements through ensemble approaches and exploring better language models for Hindi OCR error correction. Also MLM BERT with other similarity functions can be explored.

REFERENCES

[1] Sankaran, N., Jawahar, C.V. (2013). Error detection in highly inflectional languages. In 2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, pp. 1135-1139. https://doi.org/10.1109/ICDAR.2013.230

[2] Smith, R. (2007). An overview of the Tesseract OCR engine. In Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, pp. 629-633. https://doi.org/10.1109/ICDAR.2007.4376991

[3] (2016). Oliver Hellwig. Indsenz OCR. ind.senz software, http://www.indsenz.com/int/index.php, accessed on August 14, 2023.

[4] (2023). WorldTDIL. eAksharaya. TDIL. https://tdil-dc.in/eocr/index.html

[5] Mathew, M., Singh, A.K., Jawahar, C.V. (2016). Multilingual OCR for Indic scripts. In 2016 12th IAPR Workshop on Document Analysis Systems (DAS), Santorini, Greece, pp. 186-191. https://doi.org/10.1109/das.2016.68

[6] Kundaikar, T., Pawar, J.D. (2020). Multi-font devanagari text recognition using LSTM neural networks. In: Luhach, A., Kosa, J., Poonia, R., Gao, X.Z., Singh, D. (eds) First International Conference on Sustainable Technologies for Computational Intelligence. Advances in Intelligent Systems and Computing, 1045. https://doi.org/10.1007/978-981-15-0029-9_39

[7] Kundaikar, T., Pawar, J.D. (2019). Multi font error correction for devanagari script. School of Sanskrit & Indic Studies, Jawaharlal Nehru University.

[8] Vinitha, V.S., Mathew, M., Jawahar, C.V. (2017). An empirical study of effectiveness of post-processing in indic scripts. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, pp. 32-36. https://doi.org/10.1109/ICDAR.2017.362

[9] Das, D., Philip, J., Mathew, M., Jawahar, C.V. (2019). A cost efficient approach to correct OCR errors in large document collections. In 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, NSW, Australia, pp. 655-662. https://doi.org/10.1109/ICDAR.2019.00110

[10] Saluja, R., Adiga, D., Chaudhuri, P., Ramakrishnan, G., Carman, M. (2017). Error detection and corrections in Indic OCR using LSTMs. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, pp. 17-22. https://doi.org/10.1109/ICDAR.2017.13

[11] Srigiri, S., Saha, S.K. (2020). Spelling correction of OCR-generated hindi text using word embedding and levenshtein distance. In: Nath, V., Mandal, J. (eds) Nanoelectronics, Circuits and Communication Systems. NCCS 2018. Lecture Notes in Electrical Engineering, 642. https://doi.org/10.1007/978-981-15-2854-5_36

[12] 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), Sydney, NSW, Australia, pp. 160-165. https://doi.org/10.1109/ICDAR.2019.00034

[13] Nguyen, T.T.H., Jatowt, A., Coustaty, M., Doucet, A. (2021). Survey of post-OCR processing approaches. ACM Computing Surveys (CSUR), 54(6): 1-37. https://doi.org/10.1145/3453476

[14] Bansal, V., Sinha, M.K. (2001). A complete OCR for printed Hindi text in Devanagari script. In Proceedings of Sixth International Conference on Document Analysis and Recognition, pp. 0800. https://doi.org/10.1109/ICDAR.2001.953898

[15] Pal, U., Kundu, P.K., Chaudhuri, B.B. (2000). OCR error correction of an inflectional Indian language using morphological parsing. Journal of Information Science and Engineering, 16(6): 903-922.

[16] Lehal, G.S., Singh, C., Lehal, R. (2001). A shape based post processor for Gurmukhi OCR. In Proceedings of Sixth International Conference on Document Analysis and Recognition, Seattle, WA, USA, pp. 1105-1109. https://doi.org/10.1109/ICDAR.2001.953957

[17] Rasagna, V., Kumar, A., Jawahar, C.V., Manmatha, R. (2009). Robust recognition of documents by fusing results of word clusters. In 2009 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, pp. 566-570. https://doi.org/10.1109/ICDAR.2009.135

[18] 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, pp. 493-500. https://doi.org/10.1145/1815330.1815394

[19] Chen, S., Liao, H. (2022). Bert-log: Anomaly detection for system logs based on pre-trained language model. Applied Artificial Intelligence, 36(1): 2145642. https://doi.org/10.1080/08839514.2022.2145642

[20] Saleh, H., Alhothali, A., Moria, K. (2023). Detection of hate speech using bert and hate speech word embedding with deep model. Applied Artificial Intelligence, 37(1): 2166719. https://doi.org/10.1080/08839514.2023.2166719

[21] Ni, P., Wang, Q. (2022). Internet and telecommunication fraud prevention analysis based on deep learning. Applied Artificial Intelligence, 36(1): 2137630. https://doi.org/10.1080/08839514.2022.2137630

[22] Xie, G., Liu, N., Hu, X., Shen, Y. (2023). Toward prompt-enhanced sentiment analysis with mutual describable information between aspects. Applied Artificial Intelligence, 37(1): 2186432. https://doi.org/10.1080/08839514.2023.2186432

[23] Riyadh, M., Shafiq, M.O. (2022). GAN-BElectra: Enhanced multi-class sentiment analysis with limited labeled data. Applied Artificial Intelligence, 36(1): 2083794. https://doi.org/10.1080/08839514.2022.2083794

[24] Shapiro, M.C. (1989). A primer of modern standard Hindi. Motilal Banarsidass Publication.

[25] (1991). The Unicode Standard. https://www.unicode.org/charts/PDF/U0900.pdf.

[26] (2022). Shivam. Hindi dataset. https://huggingface.co/datasets/shivam/hindi_pib_processed.

[27] (2020). debiman. Img. Debiman. https://manpages.debian.org/testing/poppler-utils/pdftoppm.1.en.html.

[28] (2022). Dov Grobgeld. Unicode text to ps. Linux. https://linux.die.net/man/1/paps.

[29] Yalniz, I.Z., Manmatha, R. (2011). A fast alignment scheme for automatic OCR evaluation of books. In 2011 International Conference on Document Analysis and Recognition, Beijing, China, pp. 754-758. https://doi.org/10.1109/ICDAR.2011.157

[30] Devlin, J., Chang, M., Lee, K., Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171-4186. https://doi.org/10.18653/v1/N19-1423

[31] Hendrycks, D., Gimpel, K. (2016). Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415. https://doi.org/10.48550/arXiv.1606.08415

[32] Sun, C., Qiu, X., Xu, Y., Huang, X. (2019). How to fine-tune BERT for text classification? In: Sun, M., Huang, X., Ji, H., Liu, Z., Liu, Y. (eds) Chinese Computational Linguistics. CCL 2019. Lecture Notes in Computer Science, 11856: 194-206. https://doi.org/10.1007/978-3-030-32381-3_16

[33] Rice, S.V., Nartker, T.A. (1996). The ISRI analytic tools for OCR evaluation. UNLV/Information Science Research Institute, TR-96, 2. https://github.com/eddieantonio/ocreval