An Empirical Study of Effectiveness of Post-processing in Indic Scripts

by

Vinitha VS, Minesh Mathew, C V Jawahar

in

14th IAPR International Conference on Document Analysis and Recognition
(*ICDAR-2017*)

Kyoto, Japan

Report No: IIIT/TR/2017/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2017

An Empirical Study of Effectiveness of Post-processing in Indic Scripts

V S Vinitha, Minesh Mathew and C V Jawahar
CVIT, KCIS
International Institute of Information Technology
Hyderabad, India, 500032
Email: vinitha.vs@research.iiit.ac.in, minesh.mathew@research.iiit.ac.in, jawahar@iiit.ac.in

***Abstract*—This paper explores the effectiveness of statistical language model (SLM) and dictionary based methods for detection and correction of errors in Indic OCR output. In SLM, we use unicode level ngrams for building the language model. We compare its performance with *akshara* level ngrams and find that *akshara* level ngrams perform better in detecting the errors when compared to unicode level ngrams. We experimentally analyze the performance of Indic OCR post-processing using dictionary method, compare the performance with English and analyze the reasons for the under-performance in Indic scripts. We use four major Indian languages for our experiments, namely Hindi, Gurumukhi, Telugu and Malayalam.**

I. Introduction and Related Work

Errors in OCR system are largely unavoidable and occur due to issues like poor-quality images, complex font etc. A post-processing system can help in improving the accuracy by using the information about the patterns and constraints in the word and sentence formation to identify the errors and correct them. Indic OCRs have trailed behind in achieving accuracy comparable to English OCRs [1], [2], which have claimed accuracy close to 99%. The errors persisting in the recognized text, after the shape classifier has done the recognition task, are handled by a post-processing module. This module uses the language information to improve the accuracy of text recognized further. First we explore the effectiveness of an SLM based error correction technique, for post-processing. We have used character (unicode) level bigram and trigram language models to find the errors in the OCR output. Since *aksharas* form a more meaningful sub-unit of a word, we evaluate the performance of SLM based method using *akshara* level ngram Model. It is often assumed that a simple dictionary method, if employed can increase the accuracy significantly. We perform experiments to understand the efficacy of dictionary based method for error correction. We analyze if the dictionary based method is able to correct the errors in Indic OCR output as well as it perform in English.

Both SLM and dictionary based methods can only detect error words which are not valid words in the language, known as non-word errors. For example, consider the sentence “Take a break”. If the OCR recognizes this sentence as “Tale a break”, both the above mentioned approaches are bound to fail as Tale is a valid word in English. These class of errors known as real word errors require context information and can be corrected by using a word level statistical language model [3]. Non-word errors form a major portion of errors in the OCR output and detecting and correcting them should itself improve the OCR error rate significantly. Hence we restrict our work to only non-word errors at present.

Most early spell checking and OCR error correction systems were based on the simple technique of using a dictionary for error correction [4]. Every word recognized is checked for its presence in a dictionary and if not present is tagged as an error word. This error word is then replaced by a suitable word which is present in the dictionary. With the advances in natural language processing (NLP) techniques, especially the use of statistical language models (SLM) and noisy channel models opened doors for alternate methods for error detection and correction [3], [5]. In [3], Tong uses letter ngrams and character confusion probabilities to find possible word candidates for replacement of an error word. A word-bigram model along with Viterbi algorithm is used to predict the correct word which would fit in the sentence. For the detection of real word errors, a trigram based noisy-channel model is employed in [5]. Packer [6] used a dictionary matching approach along with variations of Hidden Markov Models (HMMs) for detecting OCR errors. In [7], Smith uses a shape classifier model, a word ngram model and a binary ngram dictionary model to detect the errors in English. In [8], the Google Web 1T 5-gram data set was used for candidate spellings generation and error correction.

The earliest works in error detection and correction were mainly done in English and other Latin languages. When the same techniques when adopted in Indic language OCRs were less successful. There have been some attempts made to develop post-processing modules for Indian languages to improve the overall OCR accuracy. The scope of applying part of speech taggers and other NLP techniques in Indic OCR post-processing is restricted due to the unavailability of such reliable models for these languages. Hence we can find most early works focusing on individual languages and their specific features such as the morphology of the words or size and shape of the words. One such attempt is a shape based post-processing system for Gurumukhi OCR [9] in which size and shape of Gurumukhi words were used to create partitions of words. This approach made use of the visual similarity between words to correct them. An error correction system for Bangla language is proposed in [10]. In this work, morphological parsing of the word is done to split the word into its root and suffix. Then a check is done to know if the root and suffix part of the word can exist together grammatically. An error detection scheme is proposed in [11], in which the author uses an SVM classifier which is trained to learn the error patterns. An ngram based error detection scheme is also employed in which each ngram is validated by using a precompiled table of ngram statistics. In a recent error detection work, [12], a bidirectional RNN based method was used to learn the patterns of correct and error words.

II. Methodology

We have conducted experiments using SLM and dictionary, to analyze the performance of post-processing in Indic scripts with English OCR output. In our experiments, we have not included shape classifier accuracy information such as the probability of recognized characters and next probable character information. We also do not take into account the character confusion information, which depends on the OCR system used for recognition. Assuming that the OCR output text is available, we use ngram probabilities to detect the errors and find possible replacement words for correcting the errors detected.

A. SLM based Post-Processing

1) Language Model Creation: The goal of using language modeling here is to learn a probability distribution over a sequence of tokens in a word. Tokens used here are characters and *aksharas* in a word [13]. In SLM based error correction, we have used the probability of bigrams and trigrams in words for error detection and correction. Bigrams provide the conditional probability of a token given the preceding token. Bigram probability is equal to the probability of their bigram, or the co-occurrence of the two tokens $P(W_{n-1}, W_n)$ divided by the probability of the preceding token. This is shown in the equation below.

$$P(W_n|W_{n-1}) = \frac{P(W_{n-1}, W_n)}{P(W_{n-1})}$$

where W_n is the n^{th} token and W_{n-1} is the token preceding it, i.e $(n-1)^{th}$ token. Similarly, a trigram uses the probability of a character, given the previous two characters in a word. We have created the language model using SRILM [14]. We have combined the corpus created from the 5K books [15] and crawled corpus [16] to create the language model. Smoothing is done to take into account those words which have not appeared in the corpora, in which case a probability of zero will be assigned to them. Many smoothing techniques are available today like Additive smoothing, Good Turing Estimate, Kneser-Ney smoothing etc [13] in which we have used the Good Turing Estimate.

2) Error Detection using SLM: In this approach, we use a unicode level language model which gives the bigram and trigram probabilities of unicodes in a word. This is used as a look-up table. We then find the average of bigram and trigram probabilities of unicodes in the input word using the look-up table. If this value is less than a threshold, we declare the word as error and correct word otherwise. The basis of this method is the assumption that all the non-word errors have a probability (calculated from its constituent ngrams) much less than that of the correct words. For each word which is fed into the error detection module, we split the word into its constituent unicode characters and obtain the bigram and trigram probabilities of its characters. If a bigram or trigram is not found in the bigram or trigram list, it is given a very low probability value, indicative of the presence of an error. The computation of the word probability is as follows. Consider a word "bags". We begin by splitting it into "b-a-g-s". We then append a start-of-word ($< s >$) and end-of-word ($< /s >$) marker at the beginning and end of word. Now we identify bigrams in the word, which in our case are $\{< s >$b, ba, ag, gs, s$< /s >$ $\}$. We find the product of these bigrams which gives bigram probability of the word. The trigrams in this case are $\{< s >$ba, bag, ags, gs$< /s >$ $\}$. We also find the trigram probability of the word. We compute the average of these two probabilities, which is then used to decide if the word is error or not. This is done by comparing the probability value with a threshold previously estimated from a list of correct words in the language.

3) Error Correction: The assumption behind this approach is that the error in a word exists at the lowest probability ngram. Now, to replace this ngram we find a list of ngrams which are at least distance from the ngram to be replaced. From the candidates for replacement, we choose that ngram which maximizes the probability of the word. Repeat the above steps on the result for a fixed number of times (for correcting multiple errors) or till average of bigram and trigram probability obtained is above a threshold. To replace an ngram, we search both these bigram and trigram list because a deletion error or an insertion error can be taken care of by looking in both the lists. There are two possible issues we face in this method. Even with many ngram replacements we may not get a word probability which is satisfactory. In this case we stop the replacement after a fixed number of iterations. The second issue is that different ngram replacements can give us different words which are all valid words in the language. In this case we choose the word whose probability is the highest.

B. Dictionary based Post-Processing

1) Dictionary Creation: Generally, the success of dictionary method depends on the size of the dictionary. In our case, to create a large dictionary in English is a fairly easy task as there are huge corpus like Google dataset [17] etc. In case of Indian languages we have the limitation of corpus availability. The only large corpus we can depend on is the crawled corpus which is abundantly available [16]. However, the variation of data you observe in books cannot be found in the crawled data. Hence, we have used the words present in 5K book corpus [15] to create a dictionary in all the five languages. The words in the test book may have overlap with words in the dictionary, but is not guaranteed. Let us call this dictionary 1. The details of the number of words in the dictionary 1 for each language is given in the Table I. We have also created another dictionary (dictionary 2) which contains all the words in dictionary 1 along with the correct words in the book used for testing.

Language	English	Hindi	Gurumukhi	Telugu	Malayalam
Words in dictionary1	38,727	92,620	90,844	258,299	331,007
Words in dictionary2	40,410	93,530	91,297	264,831	336,013

TABLE I: Details of the vocabulary size used to build the dictionary1 and dictionary2.

2) Error Detection: For error detection, we check if the word is present in the dictionary or not. The word will be labeled as a correct word only if the word is present in the dictionary. Even if the word is a valid word, if it is not present in the dictionary, it will be labeled as an error word. This is critical in case of dictionary method because a dictionary with insufficient word coverage will create a lot of false positives (correct words recognized as errors). These words when passed to the next stage will result in these words being replaced. This causes the word error rate of the OCR pipeline to increase rather than decrease after using a post-processor. The issue in Indian languages is that the dictionary cannot cover all the words. We cannot even guarantee that the dictionary will cover most of the common words in the language. The reason being that the words in languages like Malayalam, Telugu etc. are agglutinative. Though their *sandhi* split words may exist, their agglutinative combinations are difficult to cover. And we cannot add all the agglutinated words to the dictionary as the list is nearly endless because of the enormous combinations of words generated.

3) Error Correction: When a word is detected as error in the previous stage, the closest word from the dictionary is used to replace it. As a first step, a list of top 'n' candidate words are retrieved from the dictionary. There are many ways to find the closest matching word in the dictionary. A popular method is using edit distance (Levenshtein distance) based distance metric [18]. In this method, the distance is the number of deletions, insertions, or substitutions required to transform the source word into the target word. Another popular metric used to find the closest matching words is Gestalt algorithm [19] which is used in spell checkers.

We have conducted the experiments in four different Indian languages namely Hindi and Gurumukhi (Indo-Aryan languages) and Telugu and Malayalam (Dravidian languages) [20]. We also compare the results of performance in these languages with English. To analyze the errors corrected, we divided the errors based on their distance from the actual word into five classes namely, errors at distance 1 to 4 and above 4. The classification of errors based on Hamming distance from the actual word is shown in Figure 1. The errors at lower Hamming distances from the correct word should be easier to correct than the ones which are at larger distances. The errors produced depend on the shape classifier used and the quality of images used for recognition. We can find that in the Figure above, English has a significant portion of errors which are above 4 distance from the actual word. This will affect the error correction process in English. These types of errors are due to faulty images or font issues which makes comparison of different OCR errors difficult. Hence, we have considered only those errors which are at a distance less than equal to 3 from the actual word for error correction.

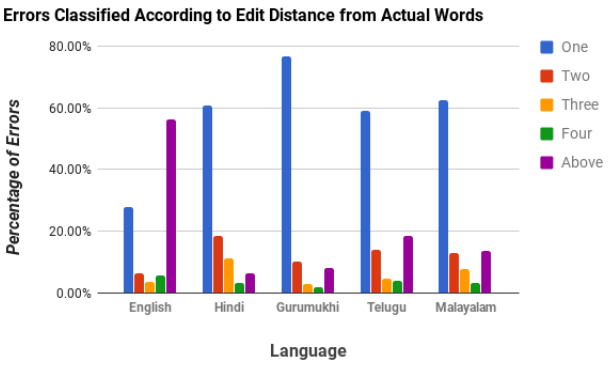

Fig. 1: Figure shows errors classified according to its Hamming distance (1 to 4 and above 4) from the actual word in different language OCR outputs.

III. Results and Analysis

A. Results using SLM

The results of error detection using unicode level SLM is shown in Figure 2. The Figure shows errors at varying distance from the actual word in different colors. It is observed that this method does a significant role in detecting errors, especially in languages like English and Gurumukhi. We repeated the experiment, this time using *aksharas* instead of unicode level SLM. The result of error detection using *aksharas* is shown in Figure 3. The error detection accuracy for errors at various

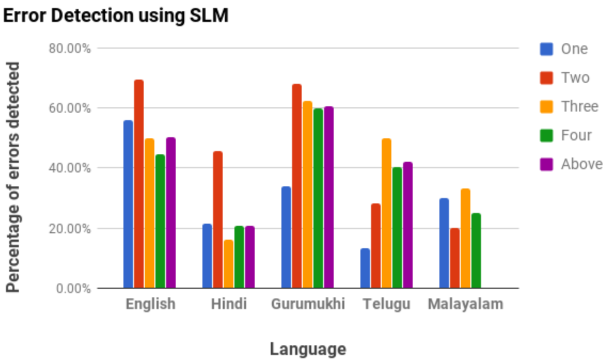

Fig. 2: Error Detection using SLM at unicode level, for different errors at varying distance from the actual word (shown in different colors).

distances are shown in different colors. A comparison of error detection performance using *akshara* and unicode is shown in Figure 4. It is clear that *akshara* level SLM do a significantly better job in detecting the errors. This is because insertion or deletion of even a small glyph in the word can alter the aksharas formed. When a valid word is split, the *aksharas* generated also will be valid. On the other hand, splitting an error word causes formation of invalid *aksharas* which are less likely to be listed in the unigram list of *aksharas*. Formation of such *aksharas* are indicative of presence of error in the word. This information is not available in unicodes; hence unicode performance is not as good as that of *akshara* split words. It is also observed that in Malayalam, more than 75% of the errors at Hamming distance 1 could be detected using *aksharas* while around 30% only could be detected using unicodes. Telugu also shows a significant improvement in error detection results when we switched to *aksharas*. The error correction using SLM at unicode level is shown in Figure 5. The error correction is not significant in any of the languages when unicode level language model is used. The result of error correction using *aksharas* is shown in Figure 6. The results using *aksharas* are better than those using unicode, both for error detection and correction. However, error correction using unicode and *akshara* do not yield promising results in any language. The use of ngrams for error correction can create multiple candidate words. Since we have to choose only one word for replacement, we have chosen the word with the highest probability. This can create a situation wherein a correct replacement which does not have the highest probability among the candidate words being ignored by the system.

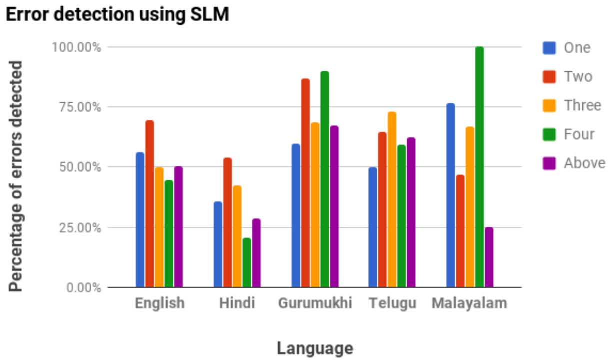

Fig. 3: Error Detection using SLM (*akshara* level for Indian languages and unicode for English) for different errors (shown in different colors) at varying distance from the actual word.

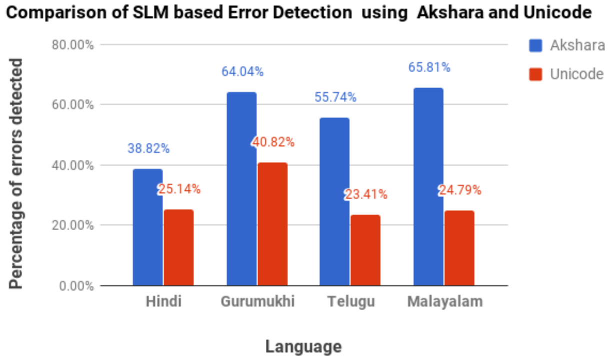

Fig. 4: Figure shows comparison of Error Detection using *akshara* (blue color) and unicode level (red color) SLM for Indian languages.

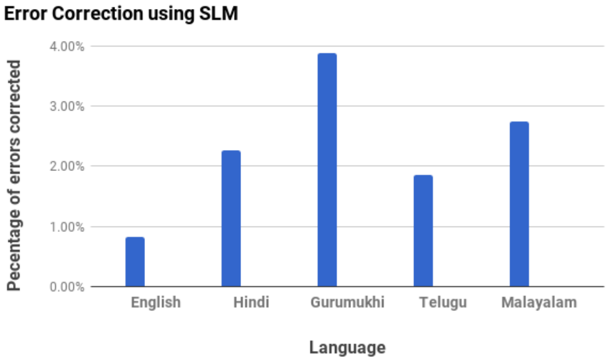

Fig. 5: Error Detection using SLM at unicode level, for different errors (shown in different colors) at varying distance from the actual word. The errors beyond distance 1 are not corrected using SLM.

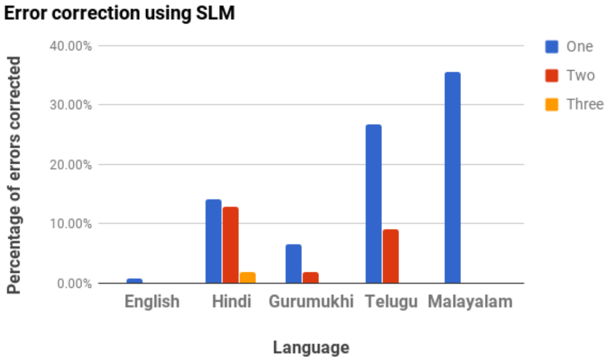

Fig. 6: Error Correction using SLM (*akshara* level for Indian languages and unicode for English) for different errors (shown in different colors) at varying distance from the actual word. Very few errors beyond distance 1 are not corrected using SLM.

B. Results and Analysis of Dictionary method

In the error detection experiment performed using dictionary, in Hindi 57% of errors were detected and in Gurumukhi 66% of the errors were detected. The highest error detection is observed in Malayalam and Telugu, 78% and 70% respectively. In English, only 44% of the errors could be detected. When we observed the errors in English, many errors which occurred were real word errors, due to incorrect recognition of punctuation etc. In Hindi, when *matras* were recognized incorrectly, inflection caused many incorrectly recognized words to be valid words. The results of experiments of error correction using dictionary method is shown in Figure 7, in which we have retrieved the top 3 candidates for error correction from the dictionary. In order to observe the performance of

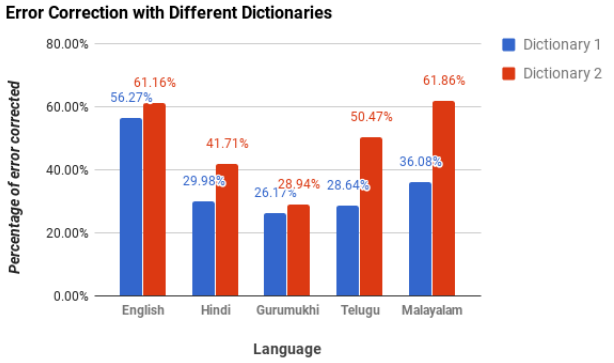

Fig. 7: Figure shows the results using 2 different Dictionaries, Dictionary which has all correct words corresponding to the error words included (red) and one in which it is not explicitly included (blue). Gestalt score is used to find the candidate words.

this method when all correct alternatives are available in the dictionary, we have done the experiment using dictionary 2. When using dictionary 1, we can see that in English, 56% of errors could be corrected. However, after using dictionary 2, the percentage of error words corrected is 61%. When we compare this with other languages, we can see that in Malayalam, the correction accuracy increased from 36% to 62%. This is a significant increase. A similar behavior is observed in Telugu, from 29% to 50%. Though Hindi and Gurumukhi also have their error correction rate improved, it is not comparable to the increase we see in Malayalam and Telugu. This shows that the dictionary 1 covers many common words in the languages in English, Hindi and Gurumukhi. Whereas in Telugu and Malayalam, many words were added which were not present in original dictionary. The error detection in inflectional languages is easy if we are able to create a good dictionary. An alternate method we can use is to split the words which are agglutinated so that the words before agglutination, if present in the dictionary can validate the word. This requires improved language processing tools in the language. The results using Levenshtein distance as the distance metric are shown in Figure 8 which is comparable to the results obtained using Gestalt score.

IV. Summary

In Indic language OCRs, traditional methods used for error detection and correction such as dictionary and character ngrams alone cannot solve the problem. A major bottleneck is the availability of a balanced corpus to create an unabridged dictionary and word level language model. The dictionary creation is particularly a difficult task for Dravidian languages such as Telugu and Malayalam due to the exploding number of unique words. We also need grammatical tools like morphological analyzers, POS taggers etc. to tackle the problem effectively. Also when compared to unicode, *aksharas* are more meaningful choice as the basic unit of a word in Indian languages. *Akshara* level language models contain more information when compared to unicode level language models.

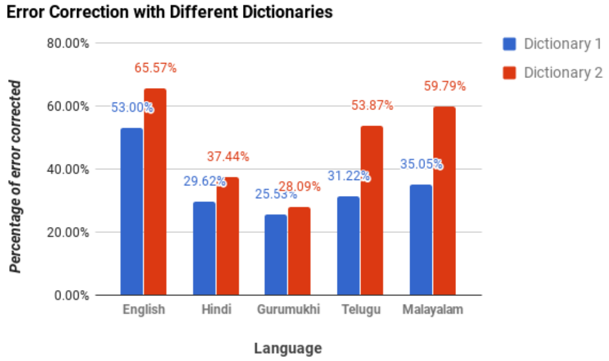

Fig. 8: Figure shows the results using 2 different Dictionaries, Dictionary which has all correct words corresponding to the error words included (red) and one in which it is not explicitly included (blue). Levenshtein distance is used to find the candidate words.

Acknowledgment. Minesh Mathew is supported by TCS Research Scholar Fellowship

References

[1] R. Smith, “An overview of the tesseract ocr engine,” 2007.
[2] “Abbyy finereader.” [Online]. Available: http://finereader.abbyy.com/
[3] X. Tong and D. A. Evans, “A statistical approach to automatic ocr error correction in context,” in *Proceedings of the fourth workshop on very large corpora*, 1996.
[4] K. Kukich, “Techniques for automatically correcting words in text,” *ACM Comput. Surv.*, 1992.
[5] A. Wilcox-OHearn, G. Hirst, and A. Budanitsky, “Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model,” 2008.
[6] T. L. Packer, “Performing information extraction to improve ocr error detection in semi-structured historical documents,” in *Historical Document Imaging and Processing*, 2011.
[7] R. Smith, “Limits on the Application of Frequency-Based Language Models to OCR,” in *ICDAR*, 2011.
[8] Y. Bassil and M. Alwani, “Ocr post-processing error correction algorithm using google online spelling suggestion,” 2012.
[9] G. Lehal, C. Singh, and R. Lehal, “A shape based post processor for Gurmukhi OCR,” in *Document Analysis and Recognition*, 2001.
[10] U. Pal, P. K. Kundu, and B. B. Chaudhuri, “OCR error correction of an inflectional indian language using morphological parsing,” *Journal Of Information Science and Engineering*, vol. 16, 2000.
[11] N. Sankaran and C. V. Jawahar, “Error detection in highly inflectional languages,” in *ICDAR*, 2013.
[12] V. Vinitha and C. Jawahar, “Error detection in indic ocrs,” in *DAS*, 2016.
[13] C. D. Manning, H. Schütze *et al.*, *Foundations of statistical natural language processing*, 1999.
[14] A. Stolcke *et al.*, “Srilm-an extensible language modeling toolkit.” in *INTERSPEECH*, 2002.
[15] D. Arya, T. Patnaik, S. Chaudhury, C. V. Jawahar, B.B.Chaudhuri, A.G.Ramakrishna, C. Bhagvati, and G. S. Lehal, “Experiences of integration and performance testing of multilingual ocr for printed indian scripts,” in *ICDAR*, 2011.
[16] D. Goldhahn, T. Eckart, and U. Quasthoff, “Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages.” in *LREC*, 2012.
[17] Y. Bassil and M. Alwani, “OCR context-sensitive error correction based on google web 1T 5-gram data set,” *American Journal of Scientific Research*, 2012.
[18] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, 1966.
[19] J. W. Ratcliff and D. E. Metzener, “Pattern-matching-the gestalt approach,” *Dr Dobbs Journal*, 1988.

[20] A. Zograf, *Languages of South Asia: a guide*, 1982.