Error Detection in Highly Inflectional Languages

Naveen Sankaran and C. V. Jawahar
International Institute of Information Technology, Hyderabad, INDIA

Abstract—**Error detection in OCR output using dictionaries and statistical language models (SLMs) have become common practice for some time now, while designing post-processors. Multiple strategies have been used successfully in English to achieve this. However, this has not yet translated towards improving error detection performance in many inflectional languages, specially Indian languages. Challenges such as large unique word list, lack of linguistic resources, lack of reliable language models, etc. are some of the reasons for this. In this paper, we investigate the major challenges in developing error detection techniques for highly inflectional Indian languages. We compare and contrast several attributes of English with inflectional languages such as Telugu and Malayalam. We make observations by analyzing statistics computed from popular corpora and relate these observations to the error detection schemes. We propose a method which can detect errors for Telugu and Malayalam, with an F-Score comparable to some of the less inflectional languages like Hindi. Our method learns from the error patterns and SLMs.**

I. Introduction

The ability to automatically detect and correct errors in OCR output has been an age old problem. Traditional methods favour using dictionary based models for error detection and correction. For example, Kukich [1] contains several methods in tackling this problem. Such methods tag a word as valid/invalid based on its presence/absence in a dictionary. The effectiveness of such methods largely depends on the size of the dictionary and the size of text corpus from which the dictionary was built. Moreover, such simple methods fail in effectively handling “out of dictionary” words. This limitation of dictionary based method resulted in the popularity of statistical language model (SLMs) for error detection.

In the past, most of the studies in error detection [2], [3] have focussed on English or very few Latin languages like German. In 1992, Kukich [1] performed experimental analysis with merely few thousands of words, while the methods discussed in 2011 by Smith [4] use a corpus as large as 100 Billion words. This growth in linguistic resources has to be attributed to the rapid proliferation of Internet, huge content (amount of text) that has become available publicly, and the increase in computing and storage powers of modern computers. Most of these studies did not have enough impact on many other languages where building and managing large dictionaries were considered to be impractical.

Although efforts have been present in developing post-processing techniques for such inflectional languages [5], [6], it has been perceived and argued that most of the attempts in English have no direct impact on such languages. There are many valid reasons for these arguments: (a) the electronic resources available for many of these languages is very small (eg. the best Telugu corpus is around 4 million words while that of the popularly used English corpus is around 100 Billion words). (b) language processing modules (like a morphological analyser, synthesizer) are still getting developed and refined. Though there exist morph analysers for many of the Indian languages, the efficiency of those analysers are nowhere close to what we have for English. They are not possibly ready to be used for error correction in OCR. (c) The unique word lists to get any meaningful results are too high to work with. We broadly agree with most of these observations. However, with significant increase in availability of newly created content for many of these languages on Internet, one can now have better estimate of the complexity of the problem. This led us in revisiting the problem.

The closest to error detection is the literature on (i) post-processor design in OCRs and (ii) spelling correction in word processing. There have been several previous works in this area. Parker *et al.* [7] used hidden markov models (HMMs) in detecting OCR errors while Golding *et al.* [8] have combined part of speech (POS) trigrams with Bayesian methods for error detection. In a recent work, Smith [4] has used a combination of classifier confidence and word nGrams in automatically detecting and correcting word errors. Error detection on inflectional languages have also been attempted before [9], [10]. In this work, we show that by properly adapting the statistical techniques, one can obtain superior performance than those reported in [9], [10] for far more complex languages. Commonly available spell checkers like GNU Aspell are also limited by the small dictionaries they use, and primitive error detection schemes. In general, SLM based methods have shown considerable success in solving this issue. In this paper, we try to understand why the problem is harder for Indic scripts from a statistical point of view. We comment on possible direction to design effective error detection and correction schemes. We also propose a method for error detection by learning error patterns from a set of examples using a SVM classifier.

II. Error Detection: Why is it Hard?

Inflectional languages have an extremely large vocabulary. Words can take forms due to the gender, sense or other contextual factors. Inflectional languages have looked at the problem of understanding words, by building word morphology analysers. However, most morphology analysers assume that the words are correct. One of our primary focus is to work with words which are erroneous, and sometime invalid. Inflectional languages are also morphologically rich and agglutinative in nature with complex structures. Besides, there are so many morphophonemic changes in the word formation process. i.e: two or more valid words can be combined to form another valid word. This issue is explained with an example in Figure 1(b). For Indian languages, the basic unit to work with is an *akshara*, which is a set of one or more consonants and a vowel. This imply that multiple Unicodes can form a single akshara. We demonstrate this formation in Fig. 1(a). An akshara could have 5 or 6 Unciodes in extreme cases.

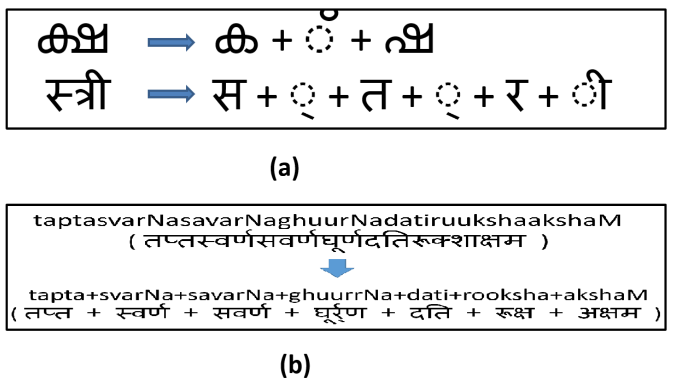

(a)
taptasvarNasavarNaghuurNadatiruukshaakshaM
(तप्तस्वर्णसवर्णघूर्णदतिरूक्शाक्षम)
tapta+svarNa+savarNa+ghuurrNa+dati+rooksha+akshaM
(तप्त + स्वर्ण + सवर्ण + घूर्ण + दति + रूक्ष + अक्षम)
(b)

Fig. 1. First image shows some sample *aksharas* and the Unicodes associated with them. As shown, there are many cases where a single akshara is formed using multiple Unicode characters, unlike English. Second figure shows a single valid word formed using combination of multiple other words.

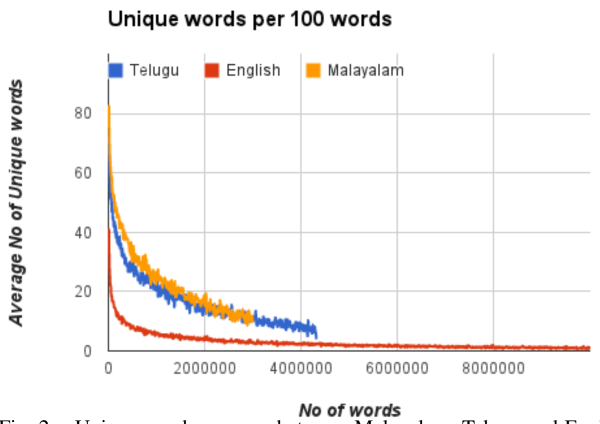


Fig. 2. Unique word coverage between Malayalam, Telugu and English. For a language to get saturated, we would require a large dataset.(better seen in colour)

A. *Size and Coverage*

We first look at the coverage of the language. Table I presents the statistics of different languages. We have considered 5 Indic scripts for our analysis and compared it with English. We have used CIIL corpus [11] for Indic scripts and have used British National Corpus (BNC) [12] for English. The BNC corpus consists of nearly 100 million words. However, to do justification in comparison (with respect to the corpus size), we have taken a subset of 5 million words from BNC to populate the data in the table. There are some interesting first level observations that we can infer from the table. The percentage of unique words in Indian languages are significantly higher than English. For eg. one in every three words in Malayalam is a new word whereas for English, it drops to one in 20 words approximately. Also note that the data shown in this table can be considered as an approximation as using a huge corpus of say 100 million can change the number. For eg. if we use the entire 100 million BNC corpus, we get a unique word count of 667,165, which is around 0.67% of the corpus. One way to gather raw text data in the present Internet age would be to crawl through the Internet and build a larger corpus. However, most of these languages doesn't have enough content available on Internet to generate a corpus which is even 10% of what we have for English. The entire Wikipedia dump of Malayalam is around a Million words in size. The case of other languages are also not very different. This table argues that the task of dictionary building is still an open task for Indic scripts as we need much larger vocabulary to cover a considerable percentage of words.

TABLE I. DATASET DETAILS FOR DIFFERENT LANGUAGES. THE PERCENTAGE OF UNIQUE WORDS FOR INDIC SCRIPTS IS MUCH LARGER WHEN COMPARING WITH ENGLISH.

Language	Total Words	Unique words	Average word length
Hindi	4,626,594	296,656 (6.42%)	3.71
Malayalam	3,057,972	912,109 (29.83%)	7.02
Kannada	2,766,191	654,799 (23.67%)	6.45
Tamil	3,763,587	775,182 (20.60%)	6.41
Telugu	4,365,122	1,117,972 (25.62%)	6.36
English	5,031,284	247,873 (4.93%)	4.66

Another factor to take into consideration is the word coverage for different languages. Word coverage tell us how many unique words are needed to cover certain percent of a language. This will help us in deciding the size of dictionary that will cover a considerable percentage of the language. There have been previous work showing such statistics [13]. We re-created the statistics from that work and the same is shown in Table II. The table shows the comparison between different languages for coverage. To cover around 80% of the words, English requires around 8K words. The same for Telugu and Malayalam are closer to 300K. A linguist may argue that this explosion is primarily due to the morphological variation, and does not really relate to the richness in the language. However, at the level of error detection, this number critically affects, until we have morphological analyzers which can work on erroneous text.

The coverage statistics for Indian scripts are actually a poor estimate of the real quantity. Figure 2, presents the variation of the unseen words per 1000 words in a language as the corpus size increases. The graph shows that while the number of unique words becomes close to zero really fast for English, for languages like Telugu and Malayalam, the convergence has still not yet happened with the available corpus. If we try to estimate the size of the corpus required to stabilize the unique word fluctuation for other languages, it would come to approximately 10M. We got the value by considering the English graph as $Ke^{-\alpha x}$ and trying to find the area under the graph to estimate for other languages. It clearly shows that while it is easy to create a dictionary for English with extremely high coverage, similar dictionaries for Malayalam or Telugu are not immediately possible.

TABLE II. WORD COVERAGE STATISTICS FOR DIFFERENT LANGUAGES. AS SHOWN THE NUMBER OF UNIQUE WORDS REQUIRED TO COVER 80PC. OF LANGUAGE VARIES FROM ONE LANGUAGE TO ANOTHER.

Corpus %	Malayalam	Tamil	Kannada	Telugu	Hindi	English
10	71	95	53	103	7	8
20	491	479	347	556	23	38
30	1969	1541	1273	2023	58	100
40	6061	4037	3593	5748	159	223
50	16,555	9680	8974	14,912	392	449
60	43,279	22,641	21,599	38,314	963	998
70	114,121	54,373	53,868	10,1110	2395	2573
80	300,515	140,164	144,424	271,474	6616	8711

B. *OCRs are Error Prone*

Multiple OCRs exist for English that can provide character level accuracies in excess of 99% [14], [15]. However even the state of the art OCR systems for many Indic scripts have sub 90% accuracy [16] at character level. This increases the possibility of having multiple errors in a word, which makes the correction tougher. Also, from Table I, we can see that the average word length is higher for Malayalam and Telugu. This means that, with a specific character classification accuracy, the word accuracies could be much lower for Malayalam and

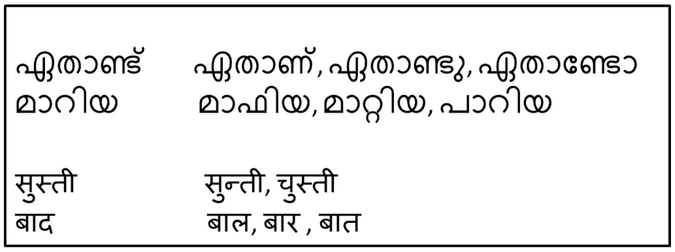
Telugu. The nature of the scripts and the complexities of the script, make the accuracies much smaller than what one could expect for English. With higher character and word error rates, the error detection and correction gets further compounded. We argue this with the help of hamming distance below.

C. Error Detection Probability

Most error detection mechanisms discuss about detecting two types of errors: “Non-Word” and “Real-Word” errors. Non-word error words are those which cannot be considered as a valid word. eg: ”Ball” getting recognized as “8a11” or “l” getting confused with “!” etc. Such errors are easy to detect as the probability of a word containing letters and digits are extremely low. The second type of errors are more tough to detect. In this category, the resultant error word is another valid word. eg: “Cat” getting recognized as “Cab”. Such errors cannot be detected by considering words in isolation as detection requires availability of larger contextual information. Usage of word bigrams or trigrams have been proven to be successful in detecting such errors for English [4].

The possibility of detecting such errors in a language requires us to know what percentage of words have the above mentioned pattern. ie. An *akshara* getting replaced by another *akshara* results in generation of another valid word. Any language which has a large percentage of words exhibiting such a pattern would present a considerable challenge in error detection as every error word can also be considered as another valid word.

We investigated the Indian languages on these lines. The results were compared against English to observe if there exists any similarity between these languages. For a given hamming distance, we decided to find out what percentage of words can generate “real-word” errors. Hamming distance measure was chosen instead of edit distance as we felt that the classifier is more likely to make a substitution error rather than insertion/deletion errors.



ഏതാണ്ട്	ഏതാണ്, ഏതാണ്ടു, ഏതാണ്ടോ
മാറിയ	മാഫിയ, മാറ്റിയ, പാറിയ
सुस्ती	सुन्ती, चुस्ती
बाद	बाल, बार , बात

Fig. 3. The above figure shows some words that can be converted to another valid character with a Hamming distance of 1.

For a given word (W) of word length (L), we computed Hamming distance (h) such that $h \leq L/2$. This was done as any word W_i of length L can be converted to W_j, which is also of length L, with a hamming distance L. eg. Any 4 letter (akshara) word can be changed into another 4 letter (akshara) word using maximum 4 substitutions. As shown in the Fig. 4, Indian scripts exhibit a very peculiar behaviour where more than 1/3rd or even 1/2 of the words can be converted to another valid word with a hamming distance of 1. While the percentage of words for English stands around 5%.

There could be multiple reasons for such characteristics to exist for Indian languages. The vocabulary size could be one

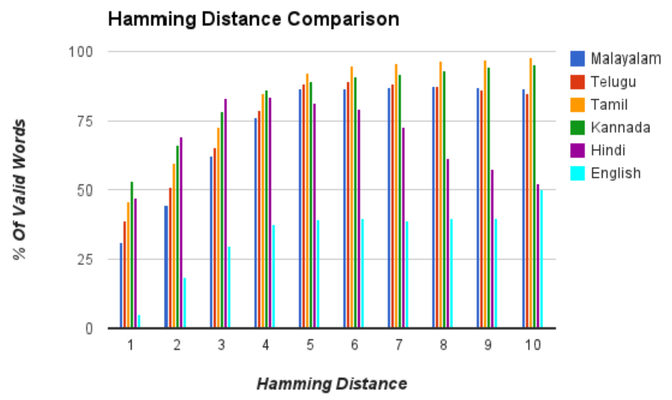


Fig. 4. The above figure shows for a given hamming distance, what percentage of words gets converted to another valid word in a language.(better seen in colour)

major reason. While English character set can be limited to 52 characters, most Indian scripts have more than 200 symbols at their disposal. Also, the presence of considerable number of ”*matras*” or vowel modifiers complicate the problem. Many such *matras* can be replaced with another *matra* to get another valid word. Figure 3 shows some examples from different languages where one word can be changed to another valid word with hamming distance one.

D. Error Correction Probability

The process of correcting a detected error is usually done by replacing the erroneous word/segment with another segment which has got the maximum probability of becoming right. The most common approach is by picking the “nearest” word with highest frequency of occurrence. To state an example, consider there is a wrong word W which has 5 words with hamming distance one in the dictionary. From these, we pick the one which has occurred the most in the language. Now the probability that the word with highest frequency of occurrence is actually the “right” word is 0.2 in this case. Infact, as the list grows larger, the difference in frequencies between the words will also start dropping. And at certain point, the significance of frequency will cease to exist. We will have a better chance of picking up the right candidate by choosing a sample randomly from the list. The odds will improve considerably when we tweak the method to include the individual nGram probability to the entire architecture. A majority voting mechanism can be used to improve the odds by selecting the word which is supported by both word and nGram dictionary. Further aid of language rules like a vowel cannot be replaced by matra can be used to improve the chances. However, this would require us to provide a set of possible matras and vowels to the algorithm.

III. Error Detection Experiments

After all the discussion about the challenges which we face in detecting errors, we look into possible methods in detecting errors. In this section, we propose multiple error detection methods and the evaluation parameter which we will be using for their evaluation.

We take Malayalam and Telugu as two working examples and demonstrate the error detection and correction methods. These languages were chosen as they appear most complex with respect to the number of unique words, average word length and word coverage.

A. Error Models

We believe that there are different types of errors that could arise and each of them needs to be handled differently. Below we give details of different error models and how it should be handled.

1) Natural erroneous data and Artificial data: These are the non-word and real-word errors mentioned in previous section. Artificial errors are the non-word errors which could be formed due to multiple reasons like degradations, noise etc. We feel that correction of such errors could be very tough using language models as the language information from such words is insufficient for any meaningful purposes. Such words would be tagged as "Wrong words" and would be left as it is. One possible approach in correcting such errors could be re-classification of such words using better features/pre-processing etc.

2) Error at akshara level: Among the real-word errors, the errors which violates the basic word building principles are easy to detect. eg: Every *akshara* would be formed by following the pattern C^*V. i.e: zero or more consonant followed by a vowel. This will help us in detecting errors like cases where a vowel occurs in between two consonants etc.

3) Errors due to confusion among aksharas: Most of the languages have got a set of aksharas which can get confused with each other. Such confusion can be taken into account while correcting errors. When presented with the decision to select one replacement from a list, higher weight can be given to such high confusion pairs ahead of other choices.

B. Methodology

We decided to look into possible solutions using both word level as well as sub-word(*akshara*) level methods.

1) Dictionary based model: The simplest of our proposed error detection models involves using a binary dictionary. Every word W_i from our test set is either considered as an in-vocabulary or as an out-of-vocabulary word. The success of such a method depends on the size of dictionary as a dictionary with larger size would infer larger word coverage.

2) nGram model: The above mentioned method was improved so that for any given word, we get the list of all possible *akshara* nGram combinations to compare it with dictionary. i.e: For a word of length 4, we get the list of all *akshara* 4 grams, 3 grams and 2 grams to be compared against the corresponding dictionary. This method possesses significant advantage against simple word dictionary based method as the possibility of detecting an error was much higher. In this case, the nGram probability of a word is shown as:

$$P\left(W_i\right) = \prod_{j=1}^{L} P\left(w_j | w_{j-n+1}^{j-1}\right) \geq \theta, \forall n : 2 \leq n \leq L$$

Here, θ is the threshold which decides if a word is valid or not. The model is flexible in a way such that we can have strict enforcement by saying criteria should be met $\forall n$ or it can be relaxed by saying if atleast one 'n' satisfies the criteria, word is considered valid.

C. Detection through learning

A formal method towards detection of errors can also be achieved using popular machine learning approaches. We use linear classification methods to classify a word as valid or invalid. We consider the nGram probabilities of a word as its features and send them to a linear classifier for learning. One constraint with such a method is that we require a set of "false" labels so that the classification can happen. For this, we use a subset of our OCR output and train a binary classifier. This classifier is used to detect errors in later stages.

D. Evaluation Metrics

Error detection performance is an important factor as improper detection can significantly reduce the accuracy while error correction. eg. tagging correct words as wrong will result in "correcting" the right word to some other word, which will decrease the overall word accuracy. We use True Positive(TP), False Positive(FP), True Negative(TN) and False Negative(FN) for evaluation purposes.

A word is considered TP if our model detects it as invalid and the result is seconded by the annotation. Similarly, a word is considered as TN when we tag it as a valid word and it is in-fact a valid word. We show these numbers in percentages where TP shows percentage of invalid words which we detected correctly among all possible invalid words present. Since our final target is to correct the words which we tag as error, TP has higher significance than FP. Also, the presence of TN will indicate the level of corruptness among the list of words which our model tagged as wrong. This is significant as higher level corruptness would indicate that an error correction module would try to "correct" a word which is already correct and thus, can make it wrong. Hence, any advantage that we gained by correcting the wrong words would be gone. We also compute Precision, Recall and F-Score based on the above values for our evaluation.

E. Experiments and Results

We test our methods on the OCR outputs of state-of-the art OCR for Malayalam and Telugu [16]. Malayalam OCR has a character accuracy of around 95% whereas for Telugu, accuracy is around 75%. Both the OCRs where used to convert around 5000 pages of printed document into unicode text. The pages are of reasonable quality, taken from multiple books. We have used a dictionary covering the entire dataset and contains around 670K words for Malayalam and 700K words for Telugu. The dictionary was generated using the CIIL corpus [11] from which all unique Malayalam/Telugu words were extracted.

We tested the approaches mentioned in previous section and evaluated them on the evaluation parameters. The results are shown in Table III and Table IV.

TABLE III. TRUE POSITIVE, FALSE POSITIVE, TRUE NEGATIVE AND FALSE NEGATIVE PERCENTAGE FOR DIFFERENT METHODS

Method	Malayalam				Telugu			
	TP	FP	TN	FN	TP	FP	TN	FN
Word Dict.	72.36	22.88	77.12	27.63	94.32	92.13	7.87	5.67
nGram	72.85	22.17	77.83	27.15	62.12	6.37	93.63	37.88
Dict+nGram	67.97	14.95	85.04	32.02	65.01	2.2	97.8	34.99
Dict+SVM	62.87	9.73	90.27	37.13	68.48	3.24	96.76	31.52

Table III shows the percentage of TP, FP, TN and FN. One observation is that the FP percentage is highest for simple dictionary based method. False positive is the percentage of words that we tagged as invalid words but are actually valid

TABLE IV. PRECISION, RECALL AND F-SCORE VALUES FOR DIFFERENT METHODS

Method	Malayalam			Telugu		
	Precision	Recall	**F-Score**	Precision	Recall	**F-Score**
Word Dict.	0.52	0.72	**0.60**	0.51	0.94	**0.68**
nGram	0.53	0.73	**0.61**	0.91	0.62	**0.73**
Dict+nGram	0.61	0.68	**0.64**	0.94	0.64	**0.76**
Dict+SVM	0.69	0.63	**0.66**	0.95	0.67	**0.78**

words. This is because any Out-of-Vocabulary (OOV) words are tagged as invalid and since the coverage of dictionary is limited, more words get tagged as invalid. This value is decreasing as we keep using nGrams or SVMs. We are able to achieve better coverage by using such methods. Percentage of false negatives shows the amount of real-word errors existing in the data. These are those words whih has been tagged as valid but is infact invalid. Such errors cannot be detected using such word/sub word based methods. Another observation is with respect to precision in Table IV. Precision provides the percentage of correctly detected invalid words from all words tagged as invalid. The significance of this measure is when we try to correct the detected errors. Low precision implies that the percentage of FP is high and the possibility of overall accuracy decreasing after error correction. This is because while correcting, we will be trying to 'correct' an already correct word which would eventually make that word invalid. Ideally, we would like to have 0 FP so that even in worst case scenario, the accuracy does not drop after error correction.

Using nGram dictionary along with word dictionary have helped in bringing down the FP ratio, especially for Telugu. However, deciding on the right threshold value determines the accuracy of this method. Dictionary coupled with SVMs have proven to be the best pick among the tested methods. Although SVMs are using the nGram features for classification, better formalization have helped in improving the results. Error detection score of 0.78 which we obtained for Telugu is comparable to detection scores for Hindi, which has detection score of 0.8.

F. Integration with an OCR

The field run of our method using an OCR for error correction is mentioned in this section. We use Malayalam OCR from [16] to obtain word outputs of 5000 pages. The OCR was having an initial accuracy of around 73% at word level. We integrated our method with OCR to detect the errors generated by it and proceeded to correct them as required. The error words where corrected by selecting the replacements from a set of possible nGrams and dictionary words. We were able to achieve a reduction of 10% in word error rate using our method.

One main limitation of our error correction mechanism is that it assumes perfect character segmentation i.e: no cuts or merges in characters. We understand this assumption not practical. While we are able to detect such errors using our method, correction of such errors would be tough using a word replacement method. This is because in cases where there are two or more characters joined together, we have to rely on edit distance and not hamming distance to find the replacement. Such a method will increase the possible number of replacement words exponentially. A better method needs to be identified for such corrections. Note that the primary objective of this paper is in exposing the challenges associated with the error detection of highly inflectional languages.

IV. CONCLUSION

In this paper, we discussed why error detection is difficult for inflectional languages, specially by comparing with languages like English. Issues related to size, coverage, hamming distance and classifier accuracies were discussed. The complexity of Indic script only add to the problems. We analysed the problem and came up with several error models and ways to detect them. Our error detection mechanisms where able to achieve an F-score of 0.66 for Malayalam and 0.78 for Telugu. Simple dictionary-frequency based error correction was performed on Malayalam and we were able to obtain word error rate reduction of 10%.

We would like to extend this work by looking into more efficient and effective methods of detecting and correcting errors. We would like to investigate the significance of specific nGrams in language that can aid us in detecting/correcting errors. A more formal method of correcting errors using machine learning techniques will also be an interesting direction to look into.

REFERENCES

[1] K. Kukich, "Techniques for automatically correcting words in text," *ACM Comput. Surv.*, vol. 24, no. 4, 1992.

[2] J. Perez-Cortes, J. Amengual, J. Arlandis, and R. Llobet, "Stochastic error-correcting parsing for OCR post-processing," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 2000.

[3] M. Wick, M. Ross, and E. Learned-Miller, "Context-sensitive Error Correction: Using Topic Models to Improve OCR," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, 2007.

[4] R. Smith, "Limits on the Application of Frequency-Based Language Models to OCR," in *ICDAR*, 2011.

[5] T. Sari and M. Sellami, "MOrpho-LEXical Analysis for Correcting OCR-generated arabic words (MOLEX)," *Ninth International Workshop on Frontiers in Handwriting Recognition*, 2002.

[6] D. V. Sharma, G. S. Lehal, and S. Mehta, "Shape encoded post processing of Gurmukhi OCR," in *ICDAR*, 2009.

[7] T. L. Packer, "Performing information extraction to improve ocr error detection in semi-structured historical documents," in *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, 2011.

[8] R. Golding and Y. Schabes, "Combining trigram-based and feature-based methods for context-sensitive spelling correction," in *Proceedings off the 34th Annual Meeting of the ACL*, 1996.

[9] B. Chaudhuri, U. Pal, and P. Kundu, "Non-word error detection and correction of an inflectional Indian Language," in *Symposium on Machine Aids for Translation and Communication(SMATAC-96)*, 1996.

[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] Central Institute Of Indian Languages (CIIL) Corpus. [Online]. Available: http://www.ciilcorpora.net/

[12] British National Corpus (BNC). [Online]. Available: http://www.natcorp.ox.ac.uk/

[13] A. Bharati, P. Rao, R. Sangal, and S. M. Bendre, "Basic Statistical Analaysis of Corpus and Cross Comparision," in *Proceedings of ICON-2002: International Conference on Natural Language Processing, Mumbai*, 2002.

[14] Tesseract Optical Character Recognition Engine . [Online]. Available: http://code.google.com/p/tesseract-ocr/

[15] Abbyy finereader. [Online]. Available: http://finereader.abbyy.com/

[16] 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 *J-MOCR Workshop,ICDAR*, 2011.