Improving Classical OCRs for Brahmic Scripts using Script Grammar Learning

Dipankar Ganguly[*], Sumeet Agarwal[*], Santanu Chaudhury[*†]
[*]Department of Electrical Engineering, Indian Institute of Technology, Delhi, India 110016
[†]Central Electronics Engineering Research Institute, Pilani, Rajasthan, India 333031
E-mail: eey147524@ee.iitd.ac.in, sumeet@ee.iitd.ac.in, schaudhury@gmail.com

***Abstract*—Classical OCRs based on isolated character (symbol) recognition have been the fundamental way of generating textual representations, particularly for Indian scripts, until the time transcription-based approaches gained momentum. Though the former approaches have been criticized as prone to failures, their accuracy has nevertheless been fairly decent in comparison with the newer transcription-based approaches. Analysis of isolated character recognition OCRs for Hindi and Bangla revealed most errors were generated in converting the output of the classifier to valid Unicode sequences, i.e., script grammar generation. Linguistic rules to generate scripts are inadequately integrated, thus resulting in a rigid Unicode generation scheme which is cumbersome to understand and error prone in adapting to new Indian scripts. In this paper we propose a machine learning-based classifier symbols to Unicode generation scheme which outperforms the existing generation scheme and improves accuracy for Devanagari and Bangla scripts.**

Keywords—Optical Character Recognition, Hidden Markov Models, Indian Language Scripts

I. Introduction

Optical Character Recognition (OCR) has been studied for many decades across various languages. Studies for OCR in Indic scripts gained popularity in the last decade [1] [2]. Though the performance of Indic script OCR's have still not matured fully and are not yet comparable to their English counterparts [3]. This can be attributed to reasons such as complexity of scripts and lack of resources. In this paper, we have used the best available solution for OCRs in Indic Scripts, analyzed it to identify the most error prone module(s) and subsequently remediate the module(s) with new approaches to improve the performance of OCRs. The experiments in this paper have been carried out for Devanagari(Hindi) and Bangla Scripts.

There have been a lot of attempts to recognize Indian scripts based on modeling the shape and intuitive features [1] [2]. The work to improve OCR output started with Bansal et al.[5] which was based on detection and correction [5] [6]. In the first work [5], they tried to use two separate dictionaries (lexicons as they describe) of root words and suffixes. Based on the grammatical matching error is identified and corrected using the dictionary. Previous work by Chaudhuri et al.[7] also used dictionary to identify and the correct errors. Though such approaches do improve accuracy but are limited to dictionary size. Scharwachter et al.[14] used Hidden Markov Models in an OCR pipeline to improve deciphering performance.

However, little work has been carried out to improve the recognition system with thorough and exhaustive error analysis, thereby working in the aftermath of identified error generating sources, with approaches to improve performance i.e. reduction in the character or word error rate. There have been few attempts to augment the OCR with Language Models which have been successful to some extent but have not explored the area of Script Grammar learning, particularly for Indic Scripts.

A. Indic Scripts

Indic scripts share a lot of similarity in terms of features and are all phonetic in nature, in the sense that they are written the way they are spoken. There is no rigid concept of "spelling" as with the western writing systems [4]. Though with geographical influences; accents change leading to variations in spellings.

Logical composition of script symbols with a common structure forms the basis of Indic Scripts, particularly the Brahmic ones; such a structure can be referred to as a "Script Grammar". This kind of structure has no counterpart in any other set of scripts in the world [4]. Indic scripts are visually composed of three zones which have been fairly exploited in designing the Classical OCRs [8]. This particular zoning seems to be very subtle way for feature engineering at segmentation and symbol recognition stages. However, this adds complexity to the Unicode generation phase of the overall system as we shall see in the subsequent sections.

B. Classical OCRs for Indic Scripts

The experiments have been conducted using Classical OCRs. The design paradigm can be broadly categorized into four stages namely (1) Pre-processing, (2) Segmentation, (3) Feature Extraction and Classification and lastly (4) Unicode generation from classifier symbols. The Pre-processing module carries out basic image processing tasks, such as Skew Correction, Binarization, etc [8]. Segmentation module works on the principles of connected components, segments the image into 3 zones: upper, lower and middle; with each zone containing literal symbols (full or partial character glyphs) [9] [10]. Feature extraction is carried out on the segmented literal symbols. Geometrical and geometry invariant (direction flagging similar to Hough features) properties of the glyphs are used for feature engineering. The extracted features based on zones are fed to a cascade of classifiers (Support Vector Machine) for symbol label prediction [2]. The symbol labels thus predicted by the classifier are then combined with linguistic rules to generate the Unicode sequences. Figure 1 shows the stages of OCR.

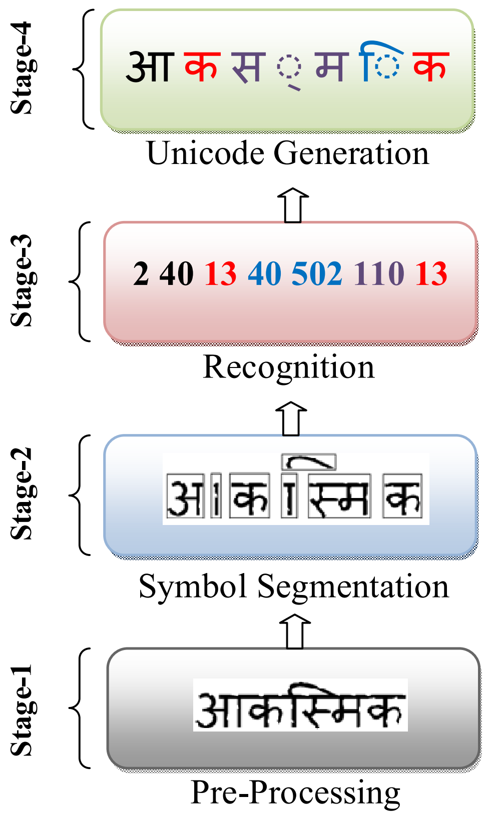


Fig. 1. Stages in OCR Pipeline used in performing experiments

II. Analysis of Error

Indic script OCRs have been analyzed on character or word error rate extensively. The confusion matrix reported also carries information about the overall accuracy of character identification [3]. However, the matrix tells nothing about the source of error, which is of prime importance for improving the recognition system. Also, the details of consonant cluster - conjuncts are completely missing, which forms a substantial part of Indic scripts. Thus, a module level analysis of the system was required to propose a new approach to reduce the error rates. The common benchmark data within Indian research community has been used for evaluation [12]. Thorough analysis of the modules revealed that the major sources of errors are in generating Unicode sequence from classifier symbols (65.5%) followed by the Segmentation module (28.6%). Similar results have also been reported previously [13]. Unicode sequence generation errors were further classified based on linguistic features as represented in Table I. Thus, it was perceived that an improvisation of Rule Based Unicode generation would certainly reduce the overall error rates.

TABLE I. LINGUISTIC CLASSIFICATION OF OCR ERRORS

Type of Error	Percentage Composition
Diacritic Error	18%
Punctuation Error	10%
Vowel Matra Error	26%
Vowel/Consonant Error	41%
Numeral (0-9) Error	5%

III. Methodology for Improvement

The Unicode generation from classifier symbols can be formulated as a sequence learning problem [13], since we are generating Unicode from labels as identified by the classifier. The existing rule based module is very cumbersome to understand. Also, the integration of linguistics features is inadequate from a debugging perspective and integration of rules is Script (Language) specific. Thus, extending rule based Unicode generation scheme to any new Indic script involves developing (from scratch) a Unicode generation module for that particular script, which is intensive and error prone. Therefore, a learned Unicode generation scheme would be best suited in this scenario, as it would be extensible and would keep the linguistic difficulties at bay.

A. *Essentials for Unicode Generation*

Once a document is to be recognized, the input image is fed into the OCR pipeline. After the two stages, the segmented symbols are available, which are separated into zones, namely – upper, middle and lower. The upper and lower zones capture the modifiers (diacritics) and the middle zone captures the complete or partial character glyph. This is then followed by zone based feature extraction and subsequent recognition by the SVM cascade (Recognition Engine). The classifier labels generated by the Recognition Engine are mapped to Unicode code characters based on handcrafted Linguistic rules. Unicode sequence can be generated from classifier labels by imposing two constraints on the learning algorithm: (1) Symbols are to be translated into valid Unicode; this is not trivial since the symbols overlap in zones and literal symbols are combinations of full, partial or combined character glyphs making it quite challenging, (2) transposition that occurs in the script needs to be adjusted for and the correct combination of the sequence needs to be learned. Many approaches can be used for solving problems with these two constraints. For this purpose, Hidden Markov Models (HMMs) have been used, as HMMs have been applied successfully to solve speech and hand writing recognition problems [12]. This sequence learning problem requirement as discussed above can be well met by the HMMs.

B. *Learning the Unicode Generation*

A significant conclusion from the discussion in the preceding section is that HMMs can be used to model Unicode sequence generation. HMMs should be able to learn a sequence/sequences of Unicode characters based on the observed states of symbol labels as generated by the SVM cascade.

The output label of the classifier is a number representing the segmented glyph. It may or may not represent a complete Unicode character. Middle zone classifier output values range from 0-300; similarly the upper and lower zone classifier symbol values is in range 0-30. Thus, upper and lower zone outputs have to be mapped to a different set to avoid overlap with the middle zone classifier outputs. The mapped values become the set of input states of HMM (360 states) and the corresponding Unicode symbols (165 states) as the set of observed states. Certain labels, for example 110, as shown in Stage 3 of Fig. 1 are responsible for generating 3 Unicode symbols – स , ्, म . Similarly, in some cases, two labels (40 and 502) combine to give a single Unicode – ि. Thus, the observed states are greater than total Unicode symbols available for Devanagri.

C. *Data Preparation For Learning*

The Indic Scripts OCRs have been developed as part of a joint effort by many participating institutes over the last decade [3]. This collaborative effort resulted in development of a state-of-art recognition system for Indian languages. The design of the system was the focus of this as a research initiative and thus, the modules have been very tightly coupled. This led to a great effort in extracting the symbol labels out of the classifier. The existing code was modified to extract classifier symbol labels from the pipeline, unaltered by the generation module. Thus, prior Unicode generation pipeline of Classical OCR shall remain unchanged for our experiments.

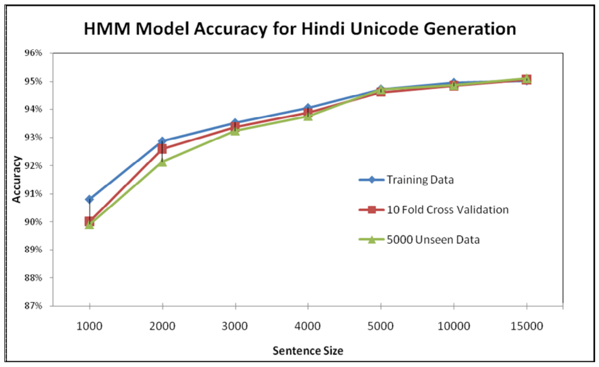


Fig. 2. HMM model accuracy for Training, Cross Validation and Unseen data acrross different training sizes for Hindi Script.

IV. Experiments, Results and Discussion

For the purpose of training and evaluation, the common benchmark data within Indian research community has been used [11]. This contains 5,000 pages of annotated data of which 20,000 sentences have been extracted. The data consisted of popular Hindi and Bengali books published in last 5 decades. Initially, we faced issues with over fitting which was overcome by correcting the zone conflicts and understanding the conjuncts label generation. Challenges faced were similar for both the scripts.

A. *Results*

The experiments were carried out with varying training size across both the scripts- Devanagri(Hindi) and Bangla as shown in Fig. 2 and Fig. 3 respectively. The graph pattern shows that a basic generation scheme is learned with very little training data, i.e. an accuracy of about 89% is achieved with 1,000 sentences in case of both the scripts. We have achieved a gain of 3.3% for Hindi and 3.1% for Bangla scripts with 5,000 sentences of unseen data from the Indian benchmark dataset as shown in Table II. The experiment has been carried with the model with highest accuracy i.e. the model trained with 15,000 sentences.

TABLE II. Comparison Of Character Error Rates For Both The Schemes

Script	**Character Error Rate**	
	Existing Rule Based Generation Scheme	***HMM Based Generation Scheme***
Hindi	8.8%	5.6%
Bangla	9.2%	6.1%

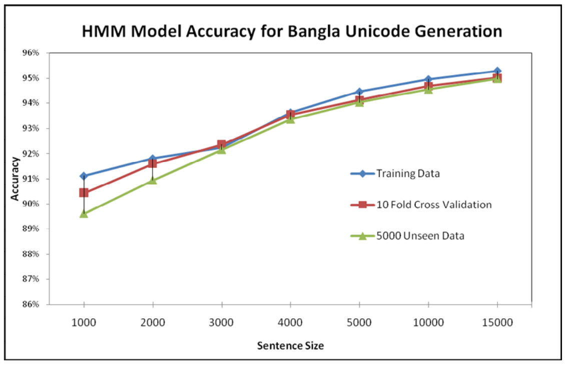


Fig. 3. HMM model accuracy for Training, Cross Validation and Unseen data acrross different training sizes for Bangla Script.

B. *Analysis of Hindi Script*

The error patterns of both the generation schemes have been analyzed to identify underlying patterns.

Error categorization has been carried out first by manually tagging each error and then by grouping similar errors into clusters for further analysis. After two iterations the categories of errors were finalized. The words with more than one error have been counted in more than one cluster.

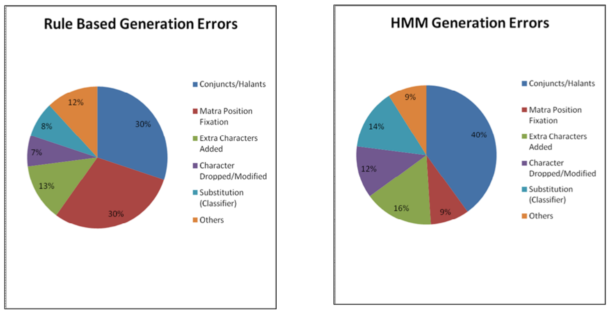


Fig. 4. Error classification in both of the generation schemes for Hindi Script.

For Hindi Script, 40% of the HMM errors constitute conjuncts generation; 30% in case of rule based generation scheme followed by the category of *Matra* position fixation as shown in Fig. 4.

Overall error composition has been shown in Fig.5, from which it can be concluded that there has been a reduction in almost all spheres of error categorization. Thus, this indicates that the HMM based formulations, excluding two cases (common pipeline), have been successful in reducing the overall error composition in each category.

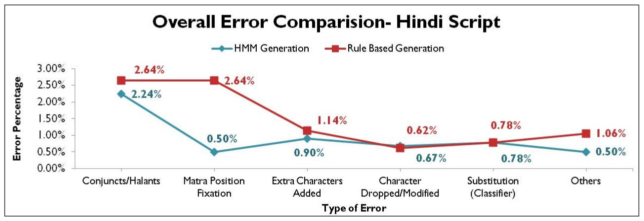
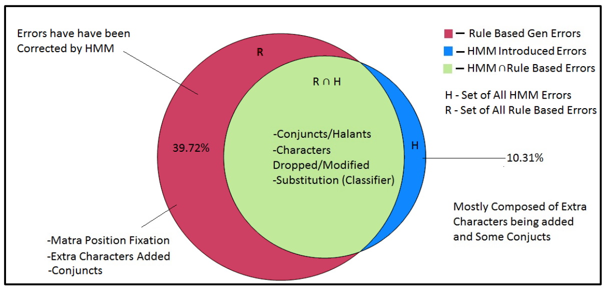


Fig. 5. Error composition under various categories of generation schemes for Hindi Script.



Errors have have been Corrected by HMM
R
R ∩ H
H
39.72%
10.31%
-Conjuncts/Halants
-Characters Dropped/Modified
-Substitution (Classifier)
— Rule Based Gen Errors
— HMM Introduced Errors
— HMM ∩Rule Based Errors
H - Set of All HMM Errors
R - Set of All Rule Based Errors
Mostly Composed of Extra Characters being added and Some Conjucts
-Matra Position Fixation
-Extra Characters Added
-Conjuncts

Fig. 6. Venn Diagram illustrating the percentages of errors that are persistent (green) , solved (red) and introduced (blue) with new generation scheme for Hindi Script.

HMM was able to correct 39.72.% of Rule based generation scheme errors mostly comprising of conjuncts, character additions and *Matra* position fixation. About 0.6% (10.31% of the total HMM errors) of new errors were introduced by HMM generation mainly consisting of rare occurring conjuncts.

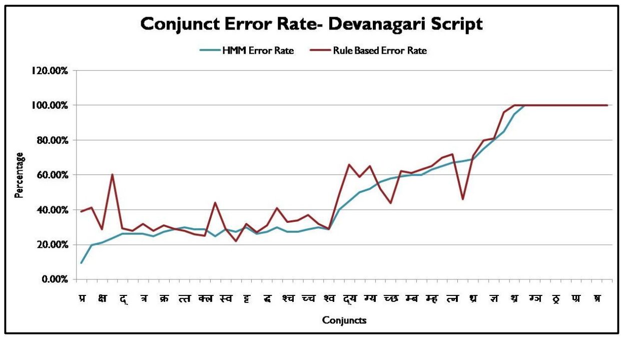


Fig. 7. Conjunct error rate on 5000 sentences of unseen data for Hindi (Devanagari) Script.

Error classification in both of the generation schemes as shown in Fig 4 indicated a necessity for further analysis, particularly for conjuncts. Fig.7 shows the error rates for conjunct generation. The conjuncts are arranged in decreasing order of frequency of occurrence in training set. A steep rise in the middle of the curve for HMM error rate implies fewer instances of those conjuncts in training data. A hybrid approach in such a scenario could be formulated to further improve the accuracy of conjunct generation.

C. *Analysis of Bangla Script*

The analysis of Bangla script revealed similar patterns with Hindi Script probably because of the similar structure of underlying Script Grammar.

Error classification analysis shows 33% percent of HMM generation errors belong to conjuncts category. The existing rule based generation scheme, on the other hand does maximum errors corresponding to *Matra* position fixation as shown in Fig 8.

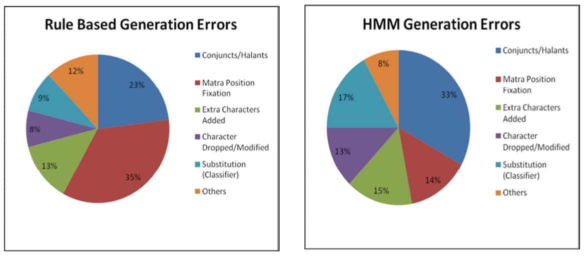


Fig. 8. Error classification in both of the generation schemes for Bangla Script.

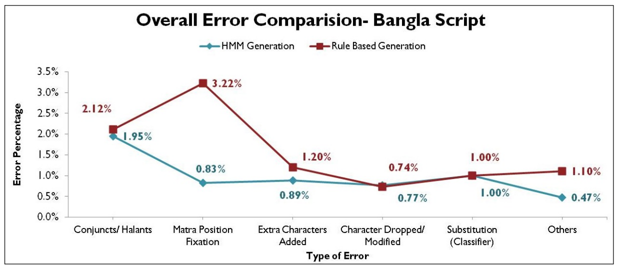


Fig. 9. Error composition under various categories of generation schemes for Bangla Script.

Overall error composition as shown in Fig.9 indicates that in comparison with Hindi, Bangla rule based generation performs better in conjuncts category. This is due to the more frequent use of conjuncts in Bangla. However, it is also evident that matra positions constitute a larger proportion in overall error contribution. There has been an improvement in 4 out of 6 categories of error classification. That justifies the overall error reduction of 3.1% in comparison with the existing generation scheme.

For Bangla script, HMM generation scheme was able to correct 44.09% of Rule based generation errors; mostly comprising of similar categories as in the case of Hindi Script. About 0.85% (13.97% of total HMM errors) of new errors were introduced by HMM generation. The same has been shown in Fig 10.

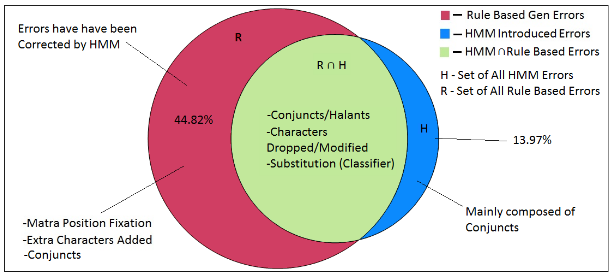

Errors have have been Corrected by HMM
R
R ∩ H
H
44.82%
-Conjuncts/Halants
-Characters Dropped/Modified
-Substitution (Classifier)
13.97%
-Matra Position Fixation
-Extra Characters Added
-Conjuncts
Mainly composed of Conjuncts
Rule Based Gen Errors
HMM Introduced Errors
HMM ∩ Rule Based Errors
H - Set of All HMM Errors
R - Set of All Rule Based Errors

Fig. 10. Venn Diagram illustrating the percentages of errors that are persistent (green) , solved (red) and introduced(blue) with new generation scheme for Bangla Script.

Analysis of conjuncts generation for Bangla was also carried out. The results are similar to that of Hindi, as shown in Fig. 11. The plot has a steep rise towards the end indicating fewer occurrences of those conjuncts in training data. An amalgamation of two techniques (rule based and HMM) taking best of both, could further improve the recognition of conjuncts.

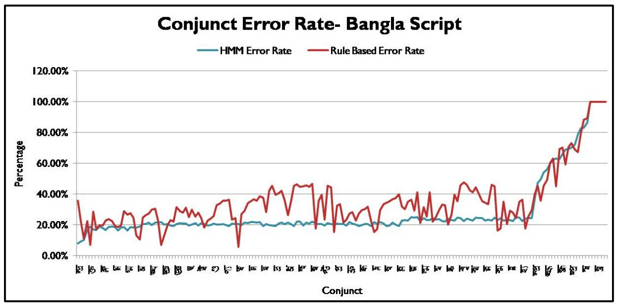


Fig. 11. Conjunct error rate on 5000 sentences of unseen data for Bangla Script.

V. Conclusion and Future Work

In this paper, we model the Unicode generation as a Script learning problem from a set of labels to Unicode for two popular Indic Scripts- Hindi and Bangla. Error analysis of existing classical OCR systems and lack of formalism in symbols to Unicode generation was the motivation behind re-engineering the existing generation module. We have used Hidden Markov Models (HMMs) to perform the desired task and used the data available within the research community to train and evaluate our approach. The current trend towards the use of recurrent neural networks, particularly BLSTM (Bidirectional Long Short Term Memory), is evident and was realized as supplementary to the discussed approach. Quantitative comparison with existing generation schemes is reported, with identification of areas for improvement. A hybrid approach of the two schemes could also be interesting to analyze. Results of two scripts have been promising and create new avenues for extending the approach to other Indic Scripts. It is expected that by adopting this approach; extending OCRs to new languages would be relatively simpler. It would be exciting to extend this approach to complex writing schemes such as Urdu, which would be different from the scripts that we have taken up in this paper.

References

[1] V. Govindaraju and S. Setlur, Guide to OCR for Indic Scripts. Springer, 2009

[2] U. Pal and B. B. Chaudhuri, “Indian Script Character Recognition: A Survey ,” in Pattern Recognition, 2004, Pages 1887-1899

[3] D. Arya, T. Patnaik, S. Chaudhury, C. V. Jawahar, B.B.Chaudhuri,A.G.Ramakrishna, C. Bhagvati, and G. S. Lehal, “ Experiences ofIntegration and Performance Testing of Multilingual OCR for PrintedIndian Scripts,” in J-MOCR Workshop,ICDAR, 2011, Pages 67-81

[4] R. Mahesh K. Sinha,"A Journey from Indian Scripts Processing to Indian Language Processing",IEEE Annals of the History of Computing Archive, Volume 31 Issue 1, January 2009 Pages 8-31

[5] V. Bansal and R. M. K. Sinha, “Partitioning and Searching Dictionary for Correction of Optically Read Devanagari Character Strings",International Journal on Document Analysis and Recognition, 2002, Volume 4, Number 4, Page 269

[6] V. Bansal and R. M. K. Sinha, “A Complete OCR for Printed Hindi Text in Devanagari Script,” in ICDAR, 2001, Pages 800-805

[7] B. B. Chaudhuri and U. Pal, “An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari (Hindi),” in ICDAR, 1997.Pages - 1011-1016

[8] Utpal Garain and B. B. Chaudhuri, “Segmentation of Touching Characters in Printed Devnagari and Bangla Scripts Using Fuzzy Multifactorial Analysis,” in ICDAR, 2001, Pages 805-809

[9] Veena Bansal and R. M. K. Sinha, “Segmentation of touching and fused Devanagari characters,” Pattern Recognition, vol. 35, no. 4, 2002. Pages 875 893

[10] Matusov, Evgeny and Kanthak, Stephan and Ney, Hermann, “On the Integration of Speech Recognition and Statistical Machine Translation,” in Proceedings of Interspeech, Lisbon, Portugal, 2005, Pages 467-474

[11] C V Jawahar and Anand Kumar, “Content-level Annotation of Large Collection of Printed Document Images,” in ICDAR, 2007, Pages 78-84

[12] Naveen Sankaran, Aman Neelappa and C. V. Jawahar,"Devanagari Text Recognition: A Transcription Based Formulation",ICDAR, 2013, Pages 678-682

[13] Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas Dietterich,"Learning Scripts as Hidden Markov Models", Proceedings of AAAI Conference on Artificial Intelligence (AAAI-2014), Pages 1565-1571

[14] Erik Scharwachter, Stephan Vogel, "Solving substitution ciphers for OCR with a semi-supervised hidden Markov model", vol. 00, no. , pp. 11-15, 2015