2011 International Conference on Document Analysis and Recognition

A Weighted Finite-State Transducer (WFST)-based Language Model for Online Indic Script Handwriting Recognition

Suhan Chowdhury and Utpal Garain
Computer Vision & Pattern Recognition (CVPR) Unit
Indian Statistical Institute
Kolkata 700108, India
utpal@isical.ac.in

Tanushyam Chattopadhyay
Innovation Lab, Kolkata
Tata Consultancy Services
Kolkata, India
t.chattopadhyay@tcs.com

Abstract— **Though designing of classifies for Indic script handwriting recognition has been researched with enough attention, use of language model has so far received little exposure. This paper attempts to develop a weighted finite-state transducer (WFST) based language model for improving the current recognition accuracy. Both the recognition hypothesis (i.e. the segmentation lattice) and the lexicon are modeled as two WFSTs. Concatenation of these two FSTs accept a valid word(s) which is (are) present in the recognition lattice. A third FST called error FST is also introduced to retrieve certain words which were missing in the previous concatenation operation. The proposed model has been tested for online Bangla handwriting recognition though the underlying principle can equally be applied for recognition of offline or printed words. Experiment on a part of ISI-Bangla handwriting database shows that while the present classifiers (without using any language model) can recognize about 73% word, use of recognition and lexicon FSTs improve this result by about 9% giving an average word-level accuracy of 82%. Introduction of error FST further improves this accuracy to 93%. This remarkable improvement in word recognition accuracy by using FST-based language model would serve as a significant revelation for the research in handwriting recognition, in general and Indic script handwriting recognition, in particular.**

Keywords-Handwrriting recognition; Language model; Finite State Transducer (FST); Indic scripts.

I. INTRODUCTION

Use of language models in OCR systems is quite common [1-4]. The success of these models for improving recognition accuracy inspired the handwriting recognition community to use similar models. Therefore, post-processing including use of language models is now a well-known topic in the field of handwriting recognition [5-13].

Handwriting recognition in Indic script has already attained considerable attention. However, the current research is still confined in finding and feature sets and designing efficient classifiers. Role of language model for Indic script OCR (printed as well as handwriting be it offline or online) is not yet explored. This paper is motivated by this need. Online handwriting recognition in Bengali (Bangla) script has been taken as a reference though the proposed model can equally be applied to printed OCR and offline handwriting recognition.

Earlier the post-processing used in handwriting recognition was mostly lexicon based [5-8]. This method maintains a series of recognition hypotheses and each hypothesis is checked and accepted against a lexicon which is normally stored in a trie structure to make the retrieval fast. The main disadvantage of this approach is its inability to correct error if none of the recognition hypotheses corresponds to a valid word. Moreover, for a given recognition hypothesis, this method may retrieve a large number of candidate words if the number of character classes is high. It takes huge memory and time. The Bengali script has a few hundred (more than 200) characters and hence, use of a simple string matching based approach is quite unattractive.

Contrary to this, n-gram based techniques are also in use [9]. In this method, character (or grapheme) bi-gram or tri-gram statistics are computed from a fixed lexicon or language corpus. Such a method can easily be applied for open-vocabulary handwriting recognition. However, problems of using an n-gram based language model are (i) many n-grams may not be encountered in training data (that leads to use of inaccurate statistics) and (ii) n-gram based model may accept lexically incorrect words.

Considering the limitations of the previous language models, this paper investigates a relative new model which is based on weighted finite-state transducers (WFSTs). WFSTs are based on the general algebraic notion of semiring [14]. The semiring abstraction permits the definition of automata representations and algorithms over a broad class of weight sets and algebraic operations. WFSTs, therefore, allow language models and recognition alternatives to be manipulated algebraically. Different models represented by WFSTs can be concatenated, unioned, intersected, composed, minimized, reversed, complemented, and transformed in a variety of other ways. In the recent past, WFSTs have emerged as a well tested technology successfully used in many tasks [15] including speech recognition, information extraction, statistical machine translation, OCR post-processing [13], etc.

WFSTs can be thought of as directed graphs whose edges are associated with input and output symbols and weights. The symbols can be Unicode characters or they can be graphemes, n-grams, or ligatures, etc. In this paper, we would like to investigate the role of WFSTs to model recognition lattice, lexicon as well as to edit the recognition hypotheses when none of the hypotheses corresponds to any word in the lexicon.

The rest of the paper is organized as follows. Section-2 describes different language models including the one based on WFST. These models were implemented and tested for Bengali handwriting recognition. Experimental results are presented in section-3. Section-4 concludes the paper.

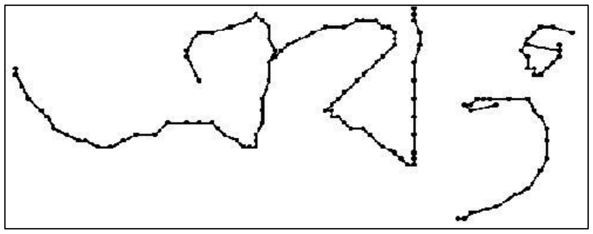

Figure 1. An online handwritten word in Bengali.

II. LANGUAGE MODELS

The input to our recognition system is a single online handwritten word as shown in Fig. 1. The word undergoes a segmentation module that normally prefers over-segmentation of a word into constituent graphemes. Figure 2 shows the segmentation of the word in Figure 1.

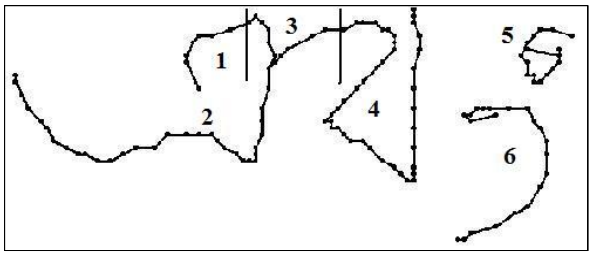

Figure 2. Segmentation of the word in Figure 1.

A MLP-based Neural Network based classifier works on this segmented data and generates recognition hypotheses. The recognizer assumes that a valid character is segmented in three or less consecutive segments. Such an empirically tested assumption reduces the number of recognition alternatives. Figure 3 shows the segmentation lattice for the word in Figure 1. The numbers on the edges correspond to the segment number as shown in Figure 2.

Each segment of this graph represents either a part of character or a complete character or a combined character sequences. Using the character classifier, each segmented part in each sequence is recognized with a confidence score. At the moment, we just know our different choices of words (i.e. character sequences). One of these choices may represent a valid word or a partially correct word. Different approaches including use of language models are used to pick the right word from these alternatives. In our study, we have explored three different approaches in order to choose the correct alternative: (i) recognition score-base approach, (ii) trie-base string matching (this is similar to using two WFSTs as discussed later), and (iii) language model based on three WFSTs.

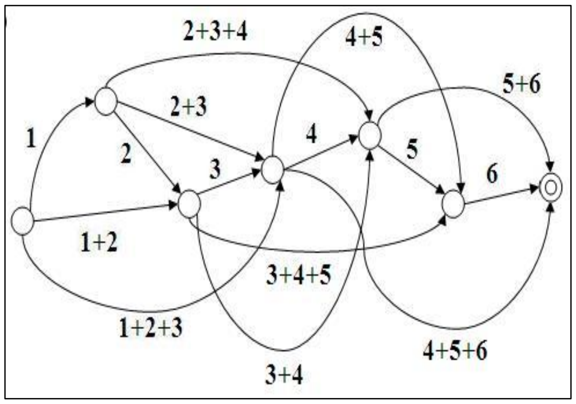

Figure 3. Segmentation Graph.

A. *Score-based approach*

Segmentation graph shows us all possible choices of word i.e. a sequence of recognized characters with some score or confidence. The path having the best score or confidence measure is selected as the best alternative. The score of a path is calculated as follows. Each recognized character in each choice holds some score; these scores are then added and divided by the number of characters along the path to get an average score for the path (or word). So each word choice has some score, maximum score help us to decide the most likely word. Note that this method does not use any language model at all rather it relies only on the character classifier. We treat such a framework a base-system which can be used for an open-vocabulary environment but it doesn't use any post-processing.

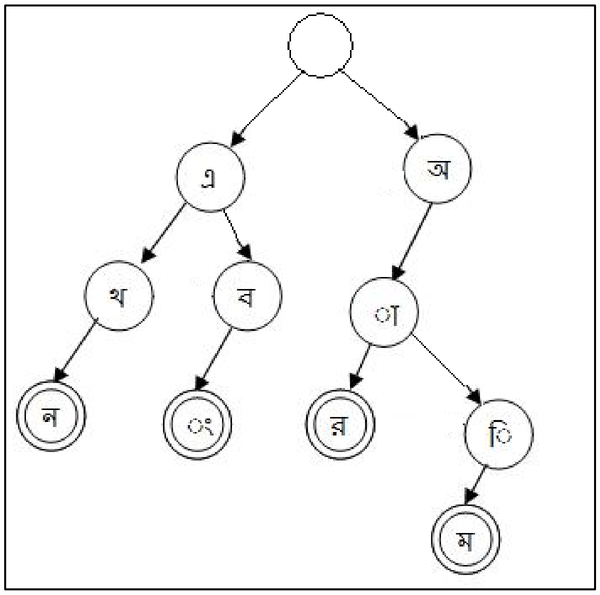

Figure 4: Lexicon represented in trie.

B. *Trie-based approach*

This is one of the lexicon based approach where the lexicon is stored in trie data structure. Figure 4 shows an example trie of four words {এখন, এবং, আর, আমি} including the one in Figure 1. Each word alternative in Figure 3 is searched in the trie and if a match is found the alternative is considered as a valid word. The use of trie structure decreases the searching complexity. If more than one word alternative is turned up as valid words, the score-base approach is then applied to choose the best alternative.

C. *FST-based Language Model*

The previous trie-based approach, in fact, can be viewed as concatenation of two WFSTs. In our study, the graph in Figure 3 is modeled as a WFST (called recognition FST or FST_R). The segment (or segments) as represented by each edge in figure 3 is considered as input in FST_R, output is the corresponding recognition output and weight is the recognition confidence. Similarly, lexicon is modeled as another FST (say, FST_L) where input and output are the same character and each weight is considered as unity. Now, composition of these two WFSTs (i.e. $FST_R \bullet FST_L$) identifies the correct word alternative (s).

The main disadvantage of this model is its inability to identify the word if no recognition alternative corresponds to any lexicon word. In such cases, the concatenation operation produces null string. Such a shortcoming motivates us to use a different model comprises of three FSTs as described next.

Use of Edit FST: In the above language model when $FST_R \bullet FST_L$ produces null, an editing FST (say, FST_E) is introduced. The idea is to edit the recognition FST in order to find the intended word from lexicon FST. For example, say xyz is the intended word which in FST_L. But no path in contains this alternative. Say, alternatives in FST_R are wyz, uvz, wz, and xv. So the composition $FST_R \bullet FST_L$ will produce null. FST_E edits members of FST_R and generates alternatives like *yz, *vz, *z, *v, w*z, u*z, w*, x*, wy*, uv*, **z, and so on where "*" denotes character whose matching is not considered. So here we use composition of three FSTs as $FST_R \bullet FST_E \bullet FST_L$. Several issues are to be considered here as explained next.

Let the input word be এখন. Using its segmented part, we get a transducer FST_R with all choices. Assume FST_R = {তখন, এথণ, এখণ, এবন, …….} and FST_L={এখন, এবং, আর, আমি}. As no word alternative corresponds to the valid word, composition of FST_R and FST_L does not produce the correct recognition. The editing FST is in form of FST_E = {*খন, এ**, এখ*, এ*ন⋯ ⋯ .} respectively. Now composition of FST_E and FST_L produces the word এখন as an alternative. Here, $FST_R \bullet FST_E \bullet FST_L$ produces only one word as output but it may produce more than one alternative as output. Consider the following example.

Say, the words আর and তার both are included in lexicon transducer, i.e. in FST_L. Let the input word be আর and no alternative in FST_R is a valid word. We get the editing FST_E in form of *◌ার (when the matching of first character is ignored). In this case, the final composition of three FSTs will produce both আর and তার as alternatives. Under this situation, the first segmented part (for which matching was ignored) is compared with these two possible characters and minimum score gives us a possible selection of the correct alternative.

III. EXPERIMENTS

To test the performance of the proposed method, we used a part of the ISI database which has been collected for online recognition of Bengali handwriting. This is indeed a huge collection from which we chose 1950 most frequent words written by 18 writers. Each subject contributes single sample for each word. Interestingly, these words contain almost all letters of the Bengali alphabet as well as the cover more than 50% of the language corpus. A MLP-based NN (Neural Network) character classifier is configured for the purpose of writer-independent word recognition. The following statistics are important for training and testing of this classifier in the context of word recognition:

Number of words (Vocabulary size): 1,950

Number of writers: 18

Total number of word samples: 35,100

Total number of character samples: 148,122

Number of character classes: 220

Number of words from which character samples used for training and validation: 28,080

Number of test words (distinct from training): 7020

Test words contain at least one sample for each of the vocabulary word. Three approaches as discussed in the previous section are tested. For implementation of the language model, we have used the open source PyOpenFST library [16] as the basis of our statistical language modeling tools. This library provides Python bindings to the original OpenFST library [17] that implements the algorithms on weighted finite-state transducers (WFSTs) and provides tools for constructing, combining, optimizing, and searching WFSTs. The word recognition results are given in Table I.

TABLE I: WORD RECOGNITION ACCURACY

#Test words = 7,020 Lexicon size = 1,950	# Correctly recognized words	(%) Correct
Score-based (No Language Model)	5,144	73.28
Trie-based Language Model	5,732	81.65
FST-based Language Model	6,528	92.99

The results reported in table-1 are quite interesting. Use of no language model gives about 73% accuracy for word recognition. Use of a trie-based language model improves this rate to about 82%. So 9% gain is observed. As this method can be viewed as composition of two FSTs, i.e. ($FST_R \bullet FST_L$), the corresponding recognition accuracy is viewed as the performance of a WSFT-based language model where both the recognition lattice and lexicon are modeled as WFSTs. Introduction of Editing FST improves the results further to about 93%. As a whole, about 20%

improvement is achieved by using the proposed language model with the base-system that doesn't use any post-processing.

Errors still remaining have been analyzed and it reveals that use of editing FST generates many alternatives in the final composition. Choosing the right word from these alternatives remains a problem that results in the errors. Moreover, use of editing FST is computationally expensive. Though for a vocabulary size of 1950 as we have used, we do not feel any processing penalty because of edit FST but for large vocabulary problem processing speed will suffer.

IV. CONCLUSIONS

A weighted finite-state transducer (WFST)-based language is configured for Indic script handwriting recognition. Online handwriting in Bengali has been taken as a reference to show the potential of the proposed language model. A gain of about 20% over the no-language model environment strongly attests the viability of the approach. The approach is script independent, environment (printed or handwritten) independent. Therefore, the same model can be trained and used for printed or handwritten (offline or online) word recognition in any language script.

This is not only the first of its kind that attempts to develop a FST-based language model for an Indic script, such a model is indeed in limited use as a post-processing tool (the paper in [13] does use a similar model). Therefore, the present research provides new avenues for doing further research on developing language models for OCR post-processing. Use of edit FST improves the accuracy but further research is needed to use it more efficiently. Use of some heuristics to limit the number of alternatives would make the present model more attractive.

REFERENCES

[1] J.J. Hull, "Incorporating Language Syntax in Visual Text Recognition with a Statistical Model," IEEE Transaction on Pattern Analysis and Machine Intelligence, 18 (12), 1251-1256, 1996.

[2] X. Tong and D.A. Evans, "A Statistical Approach to Automatic OCR Error Correction in Context." In Proc. of the 4th Workshop on Very Large Corpora (WVLC), 88-100, 1996.

[3] M. Nagata, "Japanese OCR error correction using character shape similarity and statistical language model," In Proc. of the 17th International Conference on Computational Linguistics, Montreal, Quebec, Canada, 1998.

[4] I. Bazzi, R. Schwartz, and J. Makhoul, "An Omnifont Open-Vocabulary OCR System for English and Arabic," IEEE Transactions on Pattern Analysis and Machine Intelligence, 21 (6), 495-504, 1999.

[5] A. L. Koerich, R. Sabourin and C.Y. Suen, "Large vocabulary offline handwriting recognition: A Survey," Pattern Analysis & Applications, 6, 97-121. Springer, 2003.

[6] A.R. Ahmad, C. Viard-Gaudin, and M. Khalid, "Lexicon-based word recognition using support vector machine and hidden markov model," In Proc. of 10th International Conference on Document Analysis and Recognition (ICDAR), 161-165, 2009.

[7] Z. Yao, X. Ding, and C. Liu, "On-line handwritten Chinese word recognition based on lexicon," In Proc. of 18th Int. Conf. on Pattern Recognition (ICPR), 320-323, 2006.

[8] M. Wuthrich, M. Liwicki, A. Fischer, E. Indermuhle, H. Bunke, G. Viehhauser, and M. Stolz, "Language model integration for the recognition of handwritten medieval documents," In Proc. of 10th Int. Conf. on Document Analysis and Recognition (ICDAR), 211-215, 2009.

[9] Q-F. Wang, F. Yin, and C-L. Liu. "Integrating Language model in handwritten chinese text recognition," In Proc. of 10th Int. Conf. on Document Analysis and Recognition (ICDAR), 1036-1040, 2009.

[10] Y. Zou, K. Yu, and K. Wang, "Continuous Chinese handwriting recognition with language model," In Proc. 11th Int. Conf. on Frontier in Handwriting Recognition, 2008.

[11] J.F. Pitrelli and A. Roy, "Creating word-level language models for large-vocabulary handwriting recognition," Int. J. on Document Analysis and Recognition (IJDAR), 2003.

[12] S. Quiniou, E. Anquetil, and S. Carbonnel, "Statistical language model for On-line handwritten sentence recognition," In Proc. 8th Int. Conf. on Document Analysis and Recognition (ICDAR), 1036-1040, 2005.

[13] J.C. Cortes, R. Llobet, J. Ramon, and N. Cerdan, "Using field Interdependence to improve Correction Performance in a Transducer-based OCR Post-processing System," In Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR), 2010.

[14] W. Kuich and A. Salomaa, "Semirings, Automata, Language," EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1986.

[15] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Proceedings of the IEEE, Vol. 86, No. 11, pp.2278-2324, 1998

[16] PyOpenFST: Python Bindings for the OpenFST Library. http://code.google.com/p/pyopenfst/

[17] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFST: A General and Efficient Weighted Finite-State Transducer Library," In Proc. of 9th Int. Conf. on Implementation and Application of Automata (CIAA), LNCS, Vol. 4783, 11-23, Prague, Czech Republic. Springer, 2007.