Fast Lexical Post-Processing on Cursive Script Recognition

Marco A. Torres, Susumu Kuroyanagi and Akira Iwata
Nagoya Institute of Technology
Department of Electrical and Computer Engineering[1]

Abstract

This paper presents a novel, fast approach to the problem of lexicon reduction. It is based on the concept of direct-addressing a table in which there is a slot of 1 bit for each component of 4 characters contained on the lexicon. In one of our experiments, the time needed for post-processing was reduced from $31sec$ to only $134ms$, without any loss on recognition accuracy. The structure proposed is very flexible, and can be used as a front end for contextual post-processing on Handwritten or Handprinted Recognition. It is ideal to be implemented on parallel computers or hardware.

1 Introduction

A Lexicon based text recognition system (handprinted or handwritten), receives as an input an image representing a word, and produces at the output the word on the lexicon that best matches that image (see fig. 1). It works as follows: The input image representing the word to be recognized is segmented and, features extracted from each segment are used by a Character Classifier to produce Character Candidates, which are combined to produce String Candidates, that are matched with a list of valid words (Lexicon); those strings not in the Lexicon are rejected (for example on fig. 1 the input image "test", produced 81 String Candidates, from which only two of them are valid words). Further processing could be applied to the remaining Word Candidates to select the most feasible to represent the input image on that particular context.

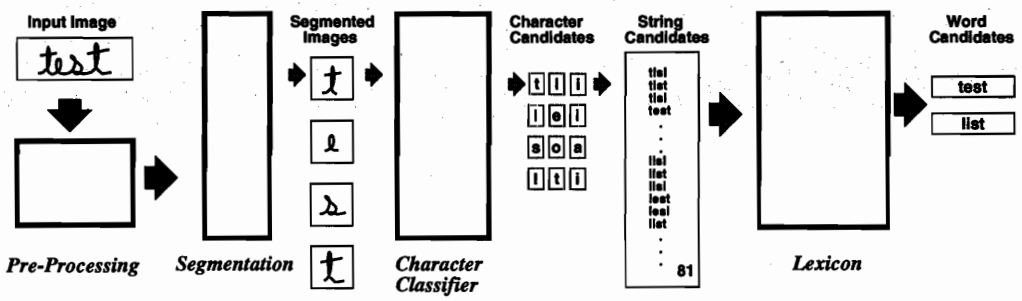

Figure 1: A Lexicon Based Text Recognition System

The time needed to analyze all the String Candidates (t_{lex}) is a function of the number of strings ($N_{strings}$) and the time needed to search in the lexicon structure for each individual string (t_{search}), $t_{lex} = t_{search} \times N_{strings}$. On the other hand, $N_{strings}$ is a function of the number of segments ($N_{segments}$) in which the input image was divided and, the

[1]Gokiso-cho, Showa-ku, Nagoya 466, Japan. e-mail: marco@mars.elcom.nitech.ac.jp

number of character candidates (N_{chars}) that are considered, $N_{strings} = (N_{chars})^{N_{segments}}$. While the list of String Candidates can be very large, it is mostly composed by not valid combinations of characters. This is specially important on script recognition where different segmentation points can be proposed and more String Candidates are produced. Then, while fast search methods are highly desirable, the most important point for efficient lexical post-processing systems is their ability to rapidly discard those not valid combinations of characters. This process is called **lexicon reduction** or **lexicon filtering**[1].

This paper examines the effectiveness of a new, fast method that reduces drastically the post-processing time by reducing the number of String Candidates. The Lexicon Reduction Method is based on pre-processing the Character Candidates before creating the actual list of String Candidates. To do this, a fast method using look-up tables (we called them Prune Tables) is proposed.

2 Description of Method

Two main goals must be considered on designing an efficient lexicon reducer: i) speed is a critical issue, especially for large lexicons, and ii) a very high percentage of the input strings must be discarded [1].

The list of String Candidates is obtained by combining the Character Candidates produced by the Classifier, for example, for an input image that was divided on 8 segments, taking 3 character candidates by segment, $3^8 = 6,561$ String Candidates will be evaluated. But, if we divide it in two groups, only $3^4 + 3^4 = 162$ components must be evaluated. Accepted components, are recombined to form the reduced list of String Candidates which will be presented to the lexicon structure on searching for valid words (see fig. 2).

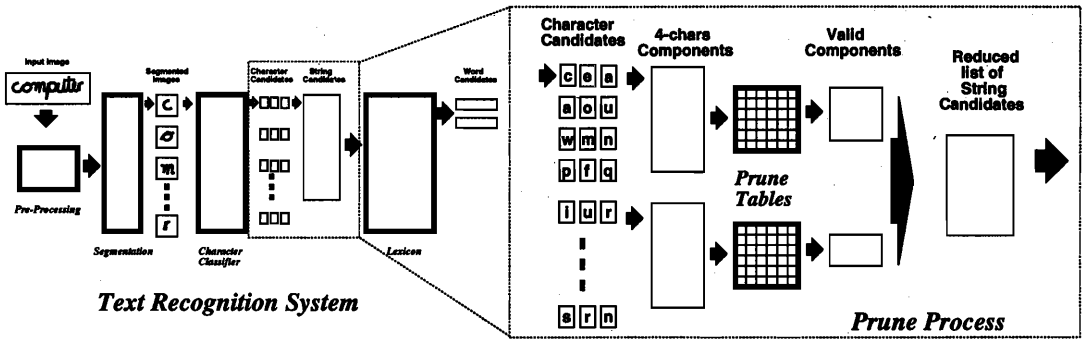

Figure 2: Including the Prune Process

If we represent $N_{strings}$ by its components:

$$N_{strings} = (N_{chars})^{\frac{N_{segments}}{2}+\frac{N_{segments}}{2}} = \underbrace{(N_{chars})^{\frac{N_{segments}}{2}}}_{component} \times \underbrace{(N_{chars})^{\frac{N_{segments}}{2}}}_{component} \quad (1)$$

Then, instead of create the entire list of String Candidates, we evaluate their components, e.g. verify if they exists on the lexicon structure or not. It is done by direct addressing [2] a table that contains one slot of 1 bit for each component in the lexicon. The size of the table is 2^r bits (where r is the size on bits of a component). Then, if we represent each of the 26 characters of English with 5 bits: from 00001 for letter **a** to

11010 for letter z, a table for 4 characters components will need 2^{20} locations of 1 bit (128KB), and for 5 characters, 2^{25}(4MB). Wells et al. [3], reported that using a long list of words, 85.4% of 2-grams are allowable, 37.1% of 3-grams would be accepted, and in the case of 4-gram only 5.5% are accepted. A good balance between a high rejection rate and memory usage could be obtained by using 4 characters components.

To produce the Prune Tables, first the lexicon is divided in four parts: 1) containing words of 1 to 4 characters, 2) words from 5 to 8 characters, 3) words from 9 to 12 characters and 4) words from 13 to 16 characters. Then each word is left justified, filling with NULL characters the gap at the right. Next, all the characters are represented on 5 bits. On this way, all the "words" from 1 to 4 characters, are represented by 20 bits (4 characters × 5 bits); then using the value of each word as the address on the Prune Table, the bit so pointed is set (bit = 1). Words from 5 to 8 characters are represented by 40 bits, then 2 Prune Tables are needed. On the same way for 9 to 12 characters, 3 tables are created, and for 13 to 16 characters, 4 tables. (see fig. 3)

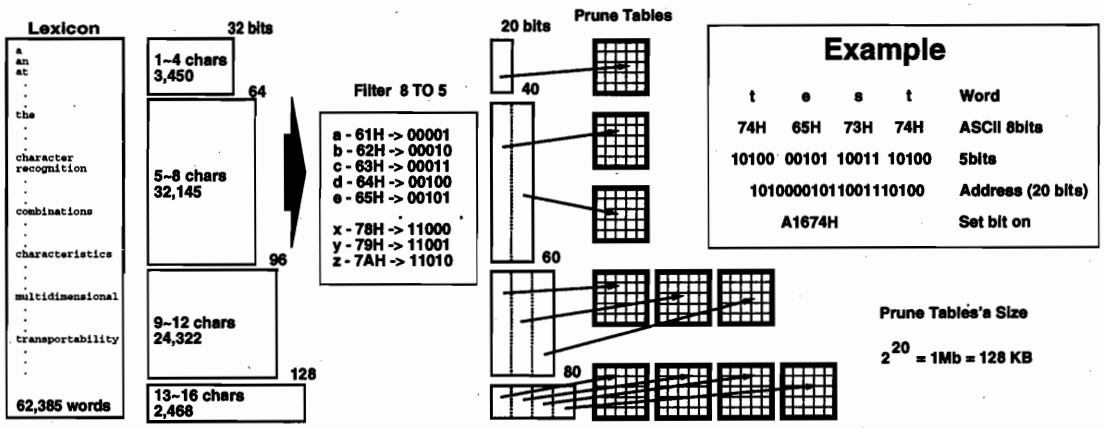

Figure 3: Prune Tables

To evaluate a component, it is only needed to read all their 4 characters, represent them on 5 bits, concatenate it to obtain the address on the Prune Table, and then verify if the bit pointed is set (accepted) or not (rejected). It is simple, efficient and fast, in all the cases it takes only one memory read.

3 Tests and Results

To evaluate the performance of the proposed method we used an English Lexicon of 62,385 words, that was divided by word size as showed on table 1. Taking 3 character candidates by segment and components of 4 characters, with Prune Tables of 128KB, two experiments were conduced:

1) Rejection Rate. Using a classifier simulator [4], we produced large data sets (see table 1) to simulate the most common mistakes produced on segmentation based recognition systems. The objective was to verify how good the proposed system can reject not valid combinations of characters.

2) Real task. Using a paragraph (see table 1), which contains words of different length. Results are illustrated on table 2.

Lexicon	
Chars	Size
1∼4	3,450
5∼8	32,145
9∼12	24,322
13∼16	2,468

Data Sets			
Name	Strings	Chars/String	Paragraph Test
the	48	3,4	The goal on text recognition is to convert information represented on a spatial form into a symbolic one. A typical text recognition system receives an image of a word as an entry, segments it and, for each segmented component produces character candidates, which are then combined to form string candidates. While the list of string candidates could be very large, it is mostly composed by not valid combinations of characters, which must be pruned quickly
alterate	62,208	8	
window	8,820	6,7,8,9	
animation	907,200	9,10,11,12	
character	139,968	9,10	
recognition	1,944,000	11, 12,13	
		75 words 40 are 4 ⇒ characters or shorter	

Table 1: Lexicon and Data sets used on the experiments

4 Discussion

Results from experiment 1, showed that the proposed system performs good on rejecting not valid components of String Candidates. From experiment 2 we confirmed that by using direct-addressing, the proposed method can achieve a very fast performance, compared with that obtained for the same task when no lexicon reduction process is used. For *Short Words* (1 to 4 characters) the proposed method can complete the Prune Process and the Search Process in ONE memory read (see fig. 4). Furthermore, on a study made by Suen [5], it was showed that most of the words found on written texts occupy two to five letters. According to that the proposed method will perform very fast on recognizing English text.

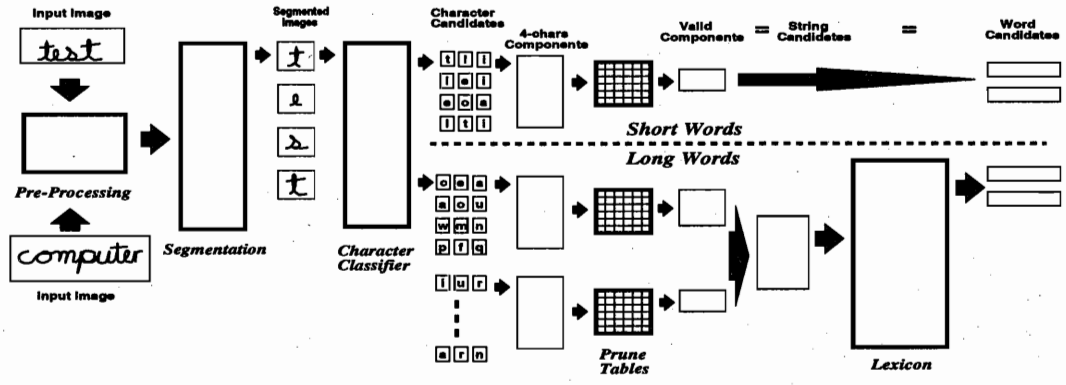

Figure 4: Text Recognition System including Prune

5 Conclusions

We presented a new method for reducing the time needed for lexical post-processing on cursive script recognition. Results obtained from our experiments showed an impressive degree of reduction without any loss on accuracy. It can be used by itself as a Search Method for words of 4 characters or less. On any case, the time needed to evaluate a 4 characters component is only one memory read. The proposed method is ideal to

Data set	Components Evaluated	Prune Time ⋆	Valid Components	String Candidates w/o Prune	String Candidates w/ Prune	Search Time ⋆ ⋆	Valid Words
Experiment 1: Degree of Rejections							
the	[48]	———	[2]	48	2	———	———
alterate	[288+181]	———	[40×19]	62,208	760	———	———
window	[153+184] †	———	[12×17] †	8,820	204 †	———	———
animation	[450+816+69]	———	[18×114×22]	907,200	45,144	———	———
character	[180+432+12]	———	[16×69×9]	139,968	9,936	———	———
recognition	[108+255+240]	———	[8×22×14]	1,944,000	2,464	———	———
Experiment 2: Real Task							
4 / 1	[3]×4	24μs	4	12			4
17 / 2	[9]×17	306μs	19	153			19
7 / 3	[27]×7	378μs	11	189			11
12 / 4	[81]×12	1,944μs	13	972			13
8 / 5	[81+3]×8	1,344μs	[9×1] [11×2] [7×2] [9×2] [13×2] [7×3] [13×1] [11×2]	1,944	145	2.9 ms	12
5 / 6	[81+9]×5	900 μs	[9×1] [13×2] [7×3] [13×4] [11×2]	3,645	130	2.6ms	7
4 / 7	[81+27]×4	864 μs	[6×1] [13×2] [7×3] [14×5]	8,748	123	2.46 ms	4
6 / 8	[81+81]×6	1,944 μs	[9×1] [13×2] [7×3][13×1] [11×2] [11×2]	39,366	113	2.26 ms	8
3 / 9	[81+81+3]×3	990 μs	[18×13×2] [9×14×3] [9×12×2]	59,049	1,062	21.2 ms	6
4 / 10	[81+81+9]×4	1,368 μs	[9×13×4] [8×6×4] [9×11×5] [16×13×4]	236,196	1,987	39.7 ms	4
4 / 11	[81+81+27]×4	1,512 μs	[16×16×4] [9×11×2] [9×12×3] [8×13×2]	708,588	1,754	35.1 ms	4
1 / 12	[81+81+81]	486 μs	[9×13×7]	531,441	819	16.4 ms	1
75 words	6,030	12.1 msⓅ			6,133	122.62 msⓈ	
Total time with Prunning Ⓟ+Ⓢ						**134.72** ms	
Total time without Prunning				1,590,303 × 20μs = **31.81** $sec.$			

⋆ The time to read a Prune Table is 2 μs. ⋆ The average time to search for a string on the lexicon structure is 20μs.
† From 153 components evaluated, only 12 were valid, and from 184 evaluated, only 17 were valid, then the number of String Candidates to be evaluated was 12 × 17 = 204.

Table 2: Results

be implemented on parallel computers, or better yet in hardware, producing a very fast system. Work is in progress to incorporate this method with a fast search method to obtain a very fast lexical post-processing system to be applied on cursive script recognition.

References

[1] S. Madhvanath and S. N. Srihari. Effective reduction of large lexicons for recognition of offline cursive script. In *Proc. of the 5th Intl. Workshop on Front. in Handwriting Rec.*, pages 189–194, 1996.

[2] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* The MIT Press, 1994.

[3] C.J. Wells, L.J. Evett, P.E. Whitby, and R.J. Whitrow. Fast dictionary look-up for contextual word recognition. *Pattern Recognition*, 23(5):501–508, 1990.

[4] R. M. K. SINHA. On using syntactic constraints in text recognition. In *Proceedings of 2nd Intl. Conf. on Doc. Anal. and Rec.*, pages 858–861, 1993.

[5] Ching Y. Suen. N-gram statistics for natural language understanding and text processing. *IEEE Trans. on Patt. Anal. and Machine Intell.*, PAMI-1, No. 2:164–172, april 1979.