Studies of Radical Model for Retrieval of Cursive Chinese Handwritten Annotations

Matthew Ma[1], Chi Zhang[2], and Patrick Wang, IAPR Fellow[3]

[1]Panasonic Information and Networking Technologies Laboratory
Panasonic Technologies, Inc. Two Research Way, Princeton, NJ 08540 USA
[2,3]College of Computer Science, Northeastern University, Boston, MA 02115

Abstract. Our research focuses on Chinese online ink matching that tries to match handwritten annotations with handwritten queries without attempting to recognize them. Previously, we proposed a semantic matching scheme that uses elastic matching with a dynamic programming approach based on the radical model of Chinese characters. By means of semantic matching, a handwritten annotation may also be retrieved independently of writers via typed text query, or stored texts can be retrieved by handwritten queries. This work concerns with the behavior of the previously proposed radical model in several aspects including character normalization, stroke segmentation, structural information, dynamic programming costs and schemes. Based on our study, a new radical model is proposed. As a result, the recall of retrieval by handwritten query reaches 90% for the first hit (an improvement of 20% over previous results) and the recall by text query reaches 80% when top 20 matches are returned.

1 Introduction and Motivation

In language computing, both on-line and off-line handwritten Chinese character recognition (HCCR) have been existing for several decades. Although online recognition has the advantage over offline because the temporal order of the input points and strokes is provided, it still has proved to be a more difficult problem than most people anticipated because of the variations of the way people write and a complex training process involved [1]. In addition, a large lexicon is to be incorporated due to the large number of characters (3,000 – 5,000) that are daily used.

Instead of handwriting recognition, some research work has been conducted on online ink matching that tries to match a handwritten query against raw ink data without attempting to recognize them [4]. This technique can be used in a document annotating and browsing system, which enables users to search their personal notes by a handwritten query. Similar work and various applications also appear elsewhere [6,7].

Recently, a semantic matching method was proposed by Ma *et al.* [5]. By extending Wang's Learning by Knowledge paradigm [8], this method focuses on the semantic approach that a human learns and recognizes things and realizes such

[1] Corresponding author. E-mail: `mma@research.panasonic.com`

approach in the matching of Chinese handwritten annotations via a radical model. The semantic matching has several advantages over previous ink matching methods [4]. First, it speeds up the existing ink matching by reducing the size of the problem. For each query, it returns only top candidates based on the matching of radicals that are extracted from handwritten annotations. The traditional raw ink matching is therefore applied only to these top candidates instead of the entire database. Secondly, only a few radicals are used thus the training process is minimized. Third, it enables the user independent retrieval without handwriting recognition. After radicals have been obtained from the raw data strings of one user, another user can type in the query by keyboard, which can be converted to radical codes immediately.

As reported in [5], the incorporation of a semantic model speeds up the matching process significantly. This is done by returning top 30 (out of 200) candidates in our experiments, consequently yielding a reduction of 80% in computation time. The drawback of semantic matching, however, is that its recall decreased from that of the original raw ink matching due to the low accuracy in radical extraction. The performance of radical extraction also affects the overall recall of retrieving handwritten annotations by typed text queries.

This work is to further study the behavior of semantic model and to improve the online Chinese ink matching results. The proposed study resulted in a new radical model for the matching of Chinese handwritten annotations. The organization of this paper is as follows. Section 2 describes several aspects of structural information in the radical model and the incorporation of such new model in radical extraction. Experiments using our new radical model on the handwritten annotation retrieval are described in Section 3. Finally, conclusions are given in Section 4.

2 Studies on Radical Model

The radical model for Chinese language is used to identify known radicals from each handwritten character and utilize these extracted radicals in the retrieval of handwritten annotations. This is called radical extraction. The drawback of the previous radical extraction is its performance. Particularly, the traditional dynamic programming was used without taking into account characteristics of Chinese language. In this section, some new aspects of Chinese radical model will be presented in order to improve the radical extraction performance.

2.1 Character Normalization and Segmentation

In this work, we employed some normalization and segmentation techniques, and experiments show they are adequate. 1) Character size normalization maybe possible once characters are successfully segmented. For simplicity, a linear normalization is used. 2) The incoming points, which are usually grouped into strokes based on the online "pen-down" and "pen-up" information, can further be segmented at local minima and maxima of the *y* values and local minima of the *x* values. We call these breaking points "internal breaking point". 3) Internal breaking points are further determined whether they are "obscure" or "obvious" depending on the degree of stroke change near it. If the change of strokes is relatively smooth around the internal breaking point, this breaking point will be considered "obscure" thus eliminated. 4) In cursive handwriting, sometimes two separate strokes are connected by an extra stroke, i.e. a connection stroke. These extra connections are not random, they are limited only to several types. In reality, the connection stroke "↖" may not appear in a handwritten character consistently. The extra connection stroke is more likely to be affected by the speed and direction of the stylus when the character was formed. Therefore, removing this extra connection stroke may reduce the effect on matching between two characters, one with connection strokes and the other without [3].

2.2 Shape Measurement

Consider a dynamic programming at stroke level. Let $C = c_1c_2...c_m$ and $R = r_1r_2...r_m$ be stroke sequences for a character and a radical, respectively. The problem of radical extraction is to take a series of operations on sequence R, from left to right, and transforms it to a subsequence of C. This can be realized by a dynamic programming procedure, in which three basic operations on strokes are defined: (a) insert a stroke, (b) delete a stroke, and (c) substitute a stroke for another. Each operation is associated with a cost. The details of dynamic programming are described elsewhere [4].

In previous implementation, stroke insertion cost and stroke deletion cost are simply in direct proportion to the length of the strokes. As for stroke substitution cost, corresponding points between two strokes are located using a separate dynamic programming procedure on point level, and Euclidean distance between each pair of two points is measured and summed. This method has two disadvantages. First, the dynamic programming on point level is time consuming. Secondly, the Euclidean distances between points can be cumulative.

Ideally, the stroke substitution measures the difference of two strokes, more precisely, the difference of their shapes. However, discrepancies exist in the current

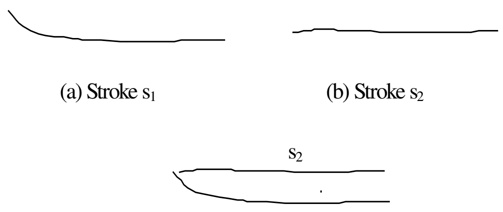


Fig. 1. Discrepancies of substitution cost based on Euclidean distance.

computation scheme. For example, as illustrated in Fig. 1, s_2 is the reference stroke, while s_1 is the stroke to be compared to s_2. In the original algorithm, before the substitution cost is computed, each stroke is temporarily shifted so that the top-left corners of the bounding boxes of the strokes are aligned (Fig. 1c). Although s_1 and s_2 are overall similar in shape except the beginning part, they will still yield a large Euclidean distance due to the deviation of the beginning part. Therefore, another method for measuring the shape similarity of two strokes based on tangent vectors is proposed.

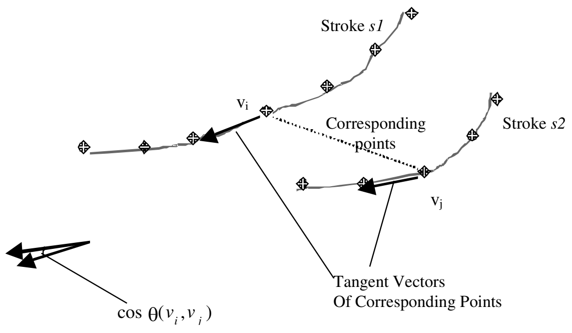


Fig. 2. Illustration of stroke measurement.

Tangent vector at a point of a stroke is defined as the vector from the current point to its next point along the stroke. Referring to Fig. 2, we define the corresponding points of two strokes as follows. Let s_1 be a stroke with l_1 points, and s_2 be a stroke with l_2 points. P_i is the ith point within s_1, the corresponding point of P_i on stroke s_2 is P_j, where $j = (i/l_1)l_2$. We calculate the substitution cost of two corresponding points P_i *and* P_j as follows:

$$point\_sub\_cost(P_i \mid s_1, s_2) = 1 - \cos\theta(v_i, v_j)$$

where v_i is the tangent vector at point P_i and v_j is the tangent vector at P_j; $\theta(v_i, v_j)$ is the angle between the two vectors, and $\theta \in [0, \pi]$. By summing up the point substitution costs for all the points along the stroke s_1, we can obtain

$$\sum_{i=1}^{l_1} point\_sub\_cost(p_i \mid s_1, s_2)$$

stroke_sub_cost(s_1,s_2), the substitution cost between stroke s_1 and s_2 as:
where l_1 is the length of stroke s_1. By further normalizing, we have

$$\max\left(\frac{l_2}{l_1}, \frac{l_1}{l_2}\right) \times \frac{(l_2 + l_1)/2}{l_1} \sum_{i=1}^{l_1} point\_sub\_cost(p_i \mid s_1, s_2)$$

Therefore, we should approximately have

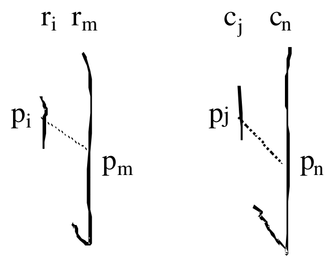
$$stroke\_sub\_cost(s_1, s_2) \approx stroke\_sub\_cost(s_2, s_1)$$

As can be seen, the new stroke substitution cost can overcome the two disadvantages mentioned earlier. First, by finding the corresponding points, we can eliminate the dynamic programming procedure in finding the pairs of corresponding points. Secondly, the calculation of substitution cost using tangent vectors does not have cumulative effects; therefore, it is a more accurate shape measurement.

2.3 Structural Information

To form a Chinese character, strokes within a character are arranged with some structural relationships (i.e. spatial relationship among strokes). Given a stroke sequence of a character alone without spatial relationships between the strokes, the character can not be determined. In this section, the stroke structural relationships embedded in Chinese language will be studied.

Center Relationships



r_i and c_j have been matched
r_m and c_n are under consideration for matching.

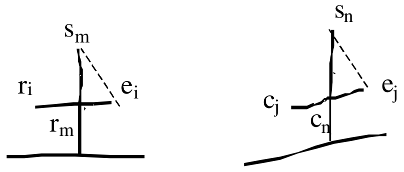
Fig. 3. Illustration of center relationships.

The weighted center of a stroke can be used to indicate the position of a stroke. The structural information can be reflected by the spatial relationship between the two stroke centers. Referring to Fig. 3, let the last two matched (substituted) strokes be r_i (the *i*th stroke of the reference radical) and c_j (the *j*th stroke of the character). The strokes currently under consideration for matching are r_m and c_n. Let p_i, p_j, p_m, p_n be the weighted centers for r_i , c_j , r_m and c_n, respectively. The vector $\overrightarrow{p_i p_m}$ reflects the spatial relationship between the two strokes r_i and r_m. Similarly, the vector $\overrightarrow{p_j p_n}$ reflects the spatial relationship between c_j and c_n. Before two strokes r_m and c_n are considered for matching, their spatial relationship with the last two matched strokes are examined:

Rule 1: If $\theta(\overrightarrow{p_i p_m}, \overrightarrow{p_j p_n}) \geq \theta_T$, r_m and c_n will not be considered for matching.

where θ_T is a threshold, currently set to $\pi/2$.

2.3.2 Starting Point and Ending Point



r_i and c_j have been matched
r_m and c_n are under consideration for matching

Fig. 4. Illustration of starting and ending point relationships.

Another important feature that reflects the structure of strokes is the relationship between ending point of a stroke and starting point of the next stroke. As illustrated in Fig. 4, let e_i and e_j be the ending points of the stroke r_i and c_j, respectively. Let s_m and s_n be the starting points of the stroke r_m and c_n, respectively. Our criteria is:

Rule 2: If $\theta(\overrightarrow{e_i s_m}, \overrightarrow{e_j s_n}) \geq \theta_T$, then r_m and c_n will not be considered for matching.

where θ_T is a threshold, currently set to $\pi/2$. Sometimes when two consecutive strokes are connected, the ending point of the first stroke happens to be the starting point of the second stroke, i.e. $e_i = s_m$ or $e_j = s_n$. In this case, the above criteria will be ignored and substitution cost for matching shall be calculated.

2.4 Categorization of Radicals

In Chinese language, the arrangement of radicals within a character is not random. For example, Cheng *et al* [1] classified the radical combinations into seven categories such as up-down (UD), left-right (LR) etc. According to the analysis of Lin *et al.* [2], over 88% of frequently used Chinese characters belong to the LR and UD types. Based on this, we categorize radicals into two main categories. In the first category, radicals start the first several strokes of a character, while in the second category, radicals end the last several strokes.

The category that a radical belongs is usually known. This category information can be reinforced into our matching process thus a wrong matching will be given a higher cost to prevent it from happening. When a reference radical is matched to a character, penalty will be added if the matched strokes within the character do not fall into the expected category. This is implemented by adjusting the cost in dynamic programming procedure.

If a first category radical is matched to a character, but substitution does not start from the first stroke of the character, all operation cost till the first substitution occurs will also be added to the total cost. Similarly, if a second category radical is not matched to the last few strokes of a character, all operation cost from the last substitution occurs till last stroke of the character will be added as penalty.

2.5 Location Similarity

By extending from the concept of radical categorization, Ma *et al.* defined radical profile and location similarity to mathematically represent the location of a radical within a character. In radical extraction scheme, location similarity gives a *non-precise* information, or, it can only *coarsely* confines the location of the radicals [5]. The dynamic programming, however, provides more accurate information of how well radical strokes are matched. We propose to use coarse information (location similarity) to sift out radicals and then use the accurate information (dynamic programming cost) to select and extract the radicals. Once radical candidates with negative location similarity are removed, the remaining radicals are ranked, according to the costs of dynamic programming, and the top two radicals with least costs are chosen as the extracted radicals.

In the previous algorithm, the total dynamic programming cost for matching a reference radical to a part of a character is the sum of all operational costs (insertion, deletion and substitution). Therefore, for each character, when all reference radicals are attempted to match to it, the radicals with fewer strokes tend to yield smaller dynamic programming costs. To solve this, we normalize the total dynamic programming cost by the length of reference radical.

2.6 Radical Code Evaluation

After radicals are extracted for each character, a character can be represented by a sequence of radical codes, i.e. radical IDs. When two characters are compared, the matching is performed using dynamic programming on a level of radical codes. Three basic operations are defined: radical insertion, radical deletion and radical substitution, each associated with an operation cost. When evaluating extracted radicals, we propose utilizing the radical extraction cost, which reflects the level of trust for extracted radicals. In implementation, extracted radicals with higher confidence (lower cost) will yield lower radical substitution cost.

3 Experiments

Our experimental data consist of three sets: 800 handwritten annotations as reference database (Set I) from four subjects, each writing 200 entries, 800 same handwritten annotations as query database (Set II) from same four subjects (written the second time), and 800 typed text as query database (Set III) corresponding to each of the 800 handwritten annotations. The experiments are conducted in three areas: the radical extraction, which is the core of semantic matching, the overall recall of searching handwritten annotations via handwritten queries and the overall recall of searching by typed text. Three methods are tested and compared with the traditional raw ink elastic matching algorithm [4]. A_0 represents the original semantic matching algorithm [5]. A_7 incorporates the new radical model in our study. A_7' is almost the same as A_7, except that the selection of top candidates returns top 15 matches, instead of top 30 matches.

Table 1 shows the total number of radicals correctly extracted from the query strings and database strings for algorithm A_0 and A_7 based on the same reference radical set. As can be seen, as a result of enhanced algorithm A_7, the number of correctly extracted radicals has increased significantly in compare to the original algorithm. And the performance gain is approximately 2 to 3 times.

Table 1. Comparison of radical extraction rate for algorithm A_0 and A_7.

	A_0	A_7
User1 data	*67*	*142*
User1 query	*58*	*137*
User2 data	*68*	*131*
User2 query	*51*	*121*
User3 data	*78*	*148*
User3 query	*61*	*144*
User4 data	*39*	*109*
User4 query	*44*	*122*

Table 2 lists the recall of the first hits for searching handwriting (Set I) with handwritten queries (Set II). As can be seen, the recall of our new algorithm A_7 has improved by *20%* for first three users, and *8%* for the 4th user. Moreover, the performance achieved by the new algorithm is very close to that of the traditional elastic matching. To compare A_7 with traditional elastic matching, because A_7 returns only top 30 candidates for final matching, it can achieve almost the same performance of original elastic matching while saves computation time by *80%*. In algorithm A_7*'*, we further reduce the computation time in half by returning only 15 top candidates. As a result, the computation time is reduced by more than 90% of the original elastic matching while achieving comparable results. In fact, it is very interesting to see that for *User2*, the matching rate of A_7 is even higher than that of the traditional elastic matching algorithm. The reason is that some interfering candidates for the traditional elastic matching algorithm has been removed from the top candidate selection process.

Table 2. Comparison of recall for first hits (searching handwriting with handwritten queries).

	A_0	A_7	A_7'	*Traditional Elastic matching*
User1	*0.725*	*0.885*	*0.845*	*0.895*
User2	*0.755*	*0.92*	*0.91*	*0.88*
User3	*0.77*	*0.96*	*0.94*	*0.98*
User4	*0.805*	*0.865*	*0.855*	*0.88*

Figure 5 shows the recall of searching by typed text queries. In this experiment, each handwritten annotation (consisting of a sequence of characters) is converted to a sequence of radical codes using radical extraction. When a text query is entered, it is immediately converted to a radical code sequence, then compared with the handwritten annotation database based on radical codes. In Fig. 5, bottom curves in each plot indicate the previous semantic matching results while the top curves stand for A_7 results. As can be seen, the matching rate for the first hit is increased by above 100%. Overall, the retrieval rate can reach *60%* with 10 top matches returned and *80%* with 20 top matches returned.

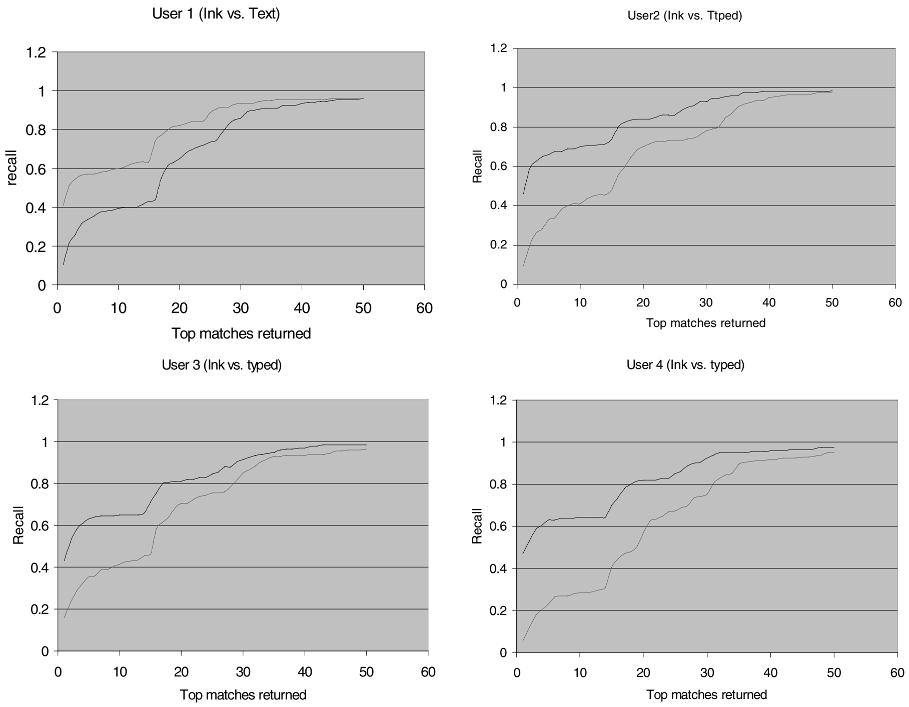


Fig. 5. The recall of searching by typed text queries. The bottom curves in each plot represent the result of A_o and the top curves represent the result of A_7.

4 Conclusions

Radical extraction plays an important role in semantic matching, in which semantics in Chinese language are incorporated early into the segmentation of handwritten annotations, and later being used to the matching of handwriting or retrieval of handwriting by typed text queries. In this work, we carefully studied and modified the radical model, based on which the radical extraction rate has increased by 100% - 200%. Several other schemes in the semantic matching network are enhanced. As a result, the recall of searching handwriting by handwritten queries has increased by

20% and reached 90% for first hits, while the computation time can be reduced by 50%. Moreover, the recall of searching by typed text queries has increased by 100% for the first hit and reached about 80% for top 20 matches returned.

The results of this work have shown great potential of semantics in the matching of Chinese handwritten annotations without full bloom handwritten recognition, in which large scale training is usually desired. To conclude, Table 3 illustrates the comparisons between various methods in the searching of Chinese annotations. It is noted that the radical model study in this work may well be extended to other languages or symbols and it is our future work.

Table 3. Comparison of handwriting matching methods.

	Speed	Performance	Handwriting Searchable	Text Searchable (user independent)
Traditional Elastic Matching	Very slow	Good	Yes	No
Previous Semantic Matching	Fast	Fair	Yes	Promising
New Semantic Matching	Very fast	Nearly good	Yes	Yes

Reference

1. F. Cheng and W. Hsu, Research on Chinese OCR in Taiwan. In Character and Handwriting Recognition, P.S.P. Wang (ed.), pp 139-164. World Scientific, 1991.
2. T.Z. Lin and K.C. Fan, Coarse classification of on-line Chinese characters via structure feature-based method. Pattern Recognition, vol. 27, pp. 1365-1377, 1994.
3. J. Liu, W.K. Cham and M.M.Y. Chang, Stroke order and stroke number free on-line Chinese character recognition using attributed relational graph matching. In Proc. 13th ICPR, pp. 259-263, August, 1996.
4. D. P. Lopresti, M. Y. Ma, P. S. P. Wang and J.D.Crisman, Ink matching of cursive Chinese handwritten annotations, Int. J. of Pattern Recognition and Artificial Intelligence, Vol.12, No.1, pp.119-141, 1998.
5. M. Y. Ma and P. S. P. Wang, Using semantics in matching cursive Chinese handwritten annotations, In Proc. SSPR'98, pp.292-301, 1998.
6. R. Manmatha, C. Han, E. M. Riseman and W. B. Croft, Indexing Handwriting Using Word Matching, Proc. of the first ACM Intl. Conf. on Digital Libraries, pp 151-159, 1996.
7. A. Poon, K. Weber and T. Cass, Scribbler: A Tool for Searching Digital Ink, Companion Proceedings of the CHI, pp. 252-253, 1995.
8. P. S. P. Wang, Learning, Representation, understanding and recognition of words - An intelligent approach, In Fundamentals in Handwriting Recognition. S. Impedovo (ed.), pp. 81-112, Springer-Verlag, 1994.
9. X.-H Xiao and R.-W Dai. Online handwritten Chinese character recognition directed by components with dynamic templates. In Proc. 17th Int. Conf. On Comp. Proc. Of Oriental Lang., pp 89-94, Hong Kong, April 1997.