Digital Ink Search Based on Character-Recognition Candidates Compared with Feature-Matching-Based Approach

Cheng CHENG(a), Nonmember, Bilan ZHU(b), Member, and Masaki NAKAGAWA(c), Fellow

SUMMARY This paper presents an approach based on character recognition to searching for keywords in on-line handwritten Japanese text. It employs an on-line character classifier and an off-line classifier or a combined classifier, which produce recognition candidates, and it searches for keywords in the lattice of candidates. It integrates scores to individually recognize characters and their geometric context. We use quadratic discriminant function(QDF) or support vector machines(SVM) models to evaluate the geometric features of individual characters and the relationships between characters. This paper also presents an approach based on feature matching that employs on-line or off-line features. We evaluate three recognition-based methods, two feature-matching-based methods, as well as ideal cases of the latter and concluded that the approach based on character recognition outperformed that based on feature matching.

key words: digital ink search, geometric context, character recognition, features matching

1. Introduction

The sales of pen-input devices such as PDAs, tablet PCs and electronic whiteboards as well as touch-sensitive devices such as iPods and iPads have been dramatically increasing and forming a new platform where keyboards are too large for operations or are not suitable for human computer interactions. Pen tips or fingertips are traced on these devices and their trajectories are expressed by a time-sequence of strokes and each stroke is again a time-sequence of coordinates from the pen being lowered to being raised. A sequence of strokes is classified as on-line handwriting or handwritten patterns in contrast to off-line handwriting patterns scanned from image scanners. On-line handwriting patterns are often called digital ink.

Digital ink is currently recognized by on-line handwritten character recognition (ONHCR) or left unrecognized. Due to the proliferation of pen-input or touch-sensitive devices, digital ink is expected to accumulate so that digital ink should be searchable even though it would not necessarily be recognized when it is written. In fact, as numerous annotations are produced on tablet PCs or iPads, they must be searched for later reference. For instance, one search is used to find the occurrences of a phrase or a keyword within digital ink. The phrase is provided with some encoding (such as ASCII or Unicode) or it is in digital ink. The problem is how to accurately and efficiently find occurrences.

Early work was carried out by Lopresti et al. [1]. They proposed ink searches at several levels of representation. They demonstrated the efficiency of prediction for the character level based on simulated text and fuzzy string matching. They also presented a matching algorithm and demonstrated its performance at the stroke level. They continued this research and formulated approximate string matching and fuzzy logic [2], which are also valid for noisy text after optical character recognition (OCR). Senda et al. presented a method of retrieving handwritten memos with handwritten queries [3]. They employed matching at the feature level. Jawahar et al. proposed a retrieval mechanism for on-line handwritten Indian text. Their proposed approach used handwriting synthesis to do matching in the ink domain as opposed to the use of a classifier [4]. Oda et al. proposed a search system for finding keywords in digital ink by employing ONHCR [5]. Zhang et al. employed a one-vs-all (OVA) trained character classifier for keyword searches from online handwritten Chinese documents [6]. The above two groups prepared candidate lattices from digital ink, which were much richer representations than just sequences of top candidates.

Digital ink searches for low-level features are generally independent of language but are often dependent on the writer while those at the level of the recognized characters are dependent on language but can be independent of the writer if ONHCR is writer-independent. Accuracy and efficiency depend on features, methods, screening, and indexing.

Searching for words or phrases within scanned documents has been studied for many years and not only for online handwriting patterns. Often target documents are comprised of very old, damaged, hand-printed or printed materials, and thus OCR does not work well on them [7]. The term “word spotting” is often used in this field.

Marukawa et al. proposed a method that could reduce the number of errors when searching from incorrect recognition results by using two or more character-recognition candidates and a confusion matrix [8]. Ota et al. extended their idea by generating search terms taking into consideration mis-segmentation and mis-recognition with corresponding confidence and bi-gram probabilities [9]. Imagawa et al. investigated the reliability of recognition results with a neural network and improved both the recall and precision rates [10]. Cao et al. described a method of key-
word spotting by modeling imperfect word segmentation as probabilities, and they integrated these probabilities into a word-spotting algorithm [11], [12]. Manmatha et al. computed similarity with a dynamic time warping (DTW) technique [13]–[17]. Konidaris et al. proposed a technique of keyword spotting in Christian manuscripts [18], [19]. Their aim was to search for typed keywords in a large collection of digitized historical printed documents in which the retrieval result was optimized by user feedback. Srihari et al. presented an effective and efficient approach to word-image matching by using gradient-based binary features [20], [21]. Bunke et al. presented a word-spotting system that employed a character classifier based on hidden Markov models [22].

Common methods and techniques can be applied between offline and online paradigms although they differ since features, suitable methods of segmentation, and methods of character recognition vary.

Focusing on an on-line paradigm, i.e., digital ink searches, we propose an ONHCR-based system and compare it with an approach of keyword searches based on feature matching. This paper is based on our previous publications [23]–[25], which have described preliminary versions of the system. The rest of this paper is organized as follows: Section 2 details the architecture for ONHCR-based digital ink searches. Section 3 describes the feature-matching-based approach. Section 4 gives details on the experiments and presents our concerns. We draw several conclusions in Sect. 5.

2. ONHCR-Based Approach

Digital ink searches refer to the task of retrieving keywords from digital ink. The architecture of our retrieval system for on-line Japanese handwritten text is outlined in Fig. 1. First, on-line Japanese handwritten texts (a sequence of strokes) are over-segmented into primitive segments according to features such as the spatial information between adjacent strokes [26]. Then, one or more consecutive primitive segments are combined to generate candidate character patterns (denoted as $c_i$), and each pattern is associated with several candidate character classes (denoted as $c_j$) with scores assigned by a character classifier [27]. The combination of all candidate patterns and character classes is represented by a segmentation and recognition candidate lattice (candidate lattice for short), as shown in Fig. 2. Last, the system searches the keyword (a sequence of character codes, denoted as $C = c_1 \ldots c_m$) in the candidate lattice, and obtains several occurrences when the resulting matching score is higher than a certain threshold. The result of digital ink search is a sequence of character patterns $X = x_1 \ldots x_m$.

Character-synchronous and time-synchronous algorithms are widely used in the field of speech recognition and character-string recognition for searching in candidate lattices. Figure 3 illustrates an example of a search algorithm for searching for a word from a candidate lattice. The character-synchronous beam search shown in Fig. 3(a) is used in our current implementation.

The algorithm works for the example shown in Fig. 3 (b). The candidate character pattern $\{1\}$ is the first node having three recognition candidates and it is followed by three succeeding patterns ($\{5\}$, $\{6\}$, and $\{7\}$). The first character of the input keyword is [$\{5\}$]. It is not in the recognition candidate classes for the pattern $\{1\}$ so that the search proceeds to the pattern $\{2\}$. Then, [$\{5\}$] appears in the candidate classes for the pattern $\{2\}$. Next, the second character [56] is compared with the pattern $\{6\}$ and the pattern $\{7\}$ following the character pattern $\{2\}$ and it is found in the candidate classes for the pattern $\{8\}$. Third, the last character [5n] is compared with the last pattern $\{9\}$ and [$\{n\}$] appears in the candidate classes for the pattern $\{9\}$. Finally, strokes from $\{2\}$, $\{8\}$ and $\{9\}$ are outputs. The search continues from the pattern $\{3\}$ but there is no more match with the keyword.

Here, precision is often affected by tiny discrepancies between strokes in digital ink. This happens when the correct sequence of strokes contains a subsequence of them, which also produces a high score, as shown in Fig. 4. Although the strokes in (A) produce the highest score (since "$\{3\}$" matches "$\{4\}$" better than "$\{2\}$"), the strokes in (B) also produce a still higher score for the input keyword. Since both (A) and (B) are output by the search, (B) reduces the search’s overall precision. This effect is especially noticeable in the Japanese language since many Japanese hiragana/katakana characters have two or three slightly different family members: one for voiceless sound, one for voiced sound, and another for the p-sound.

We use a straightforward and efficient solution to this problem by applying a character classifier to the candidate lattice. The classifier is trained on a large collection of handwritten Japanese text and produces a score for each character in the candidate lattice. The scores are then used to rank the candidate lattice and select the most likely characters.

**Fig. 1** Flow for ONHCR-based keyword search.

**Fig. 2** Segmentation and recognition candidate lattice (candidate character classes are associated with each candidate character pattern).
CHENG et al.: DIGITAL INK SEARCH BASED ON CHARACTER-RECOGNITION CANDIDATES COMPARED WITH FEATURE-MATCHING-BASED APPROACH

Algorithm: Search algorithm
Input: Keyword $C = c_1...c_m$, candidate character patterns $X=x_1...x_n$
Output: a sequence of character patterns $x_{i+1}...x_{jm}$

1. Begin: Initialize: $i=1, j=1$
2. for $i=1$ to $n$
3. 
4. Match($c_i, x_i$)
5. if $c_i$ is a character class of $x_i$
6. label the character pattern $x_i$
7. if $c_i$ is the last character of keyword $x_i$
8. go to line 2
9. else
10. $x_{i} =$ the succeeding character pattern of $x_i$
11. set $j=1$
12. end

(a) Pseudocode for search algorithm

Fig. 3 OHHCR-based search algorithm and example of path search.

(b) Path search for keyword (each dotted vertical line denotes a segmentation point)

Fig. 4 Correct stroke sequence (A) and subsequence (B) giving high scores (each dotted vertical line denotes a segmentation point).

2.1 Similarity Measure

Given a keyword, the system searches in the candidate lattice to find paths (stroke sequences) matching the input keyword. There is some noise in the output paths. Therefore, it is important for the method of searching in the candidate lattice to reduce the amount of computation and search noise. We design a linear discriminant function to evaluate the goodness of paths such that the correct path has the highest score to distinguish correct retrieval results from incorrect ones in the candidate lattice. The discriminant function for retrieval result is defined as

$$f(X, C) = \frac{\sum_{i=1}^{6} f_i + \lambda_i + \lambda_0}{n}$$

(1)

where $F = [f_1, f_2...f_6]$ is the feature evaluation functions where reflect the geometric context. Here, $f_6$ incorporates the score of character recognition, $\lambda = [\lambda_1, \lambda_2...\lambda_6]$ is a set of weighting parameters, and $\lambda_0$ allows for any fixed offset, which is sometimes called a bias parameter. The $n$ is the length of the input keyword.

Table 1 summarizes the six functions to evaluate features in Eq. (1), $n$ is the length of the input keyword, $c_i$ is the $i$-th character code of the input keyword, and $x_i$ denotes the $i$-th character pattern matching with $c_i$.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1(x_i, x_{i+1}, c_i, c_{i+1})$</td>
<td>QDF function for binary geometric feature vector $p^b_i$ between $x_i$ and $x_{i+1}$ with character classes $c_i$, $c_{i+1}$. The $p^b_i$ has two elements measured from the bounding boxes of two adjacent character patterns $x_i, x_{i+1}$, i.e., the vertical distances between the upper bounds and between the lower bounds shown in Fig. 5.</td>
</tr>
<tr>
<td>$HW(x_i, c_i)$</td>
<td>QDF function for geometric inner gap feature vector $q_i$ of $x_i$ and character class $c_i$. The $q_i$ comprises the six values in Fig. 6. The first three values represent the horizontal gaps of three vertical slits (partitioned from vertical projection), and the last three represent the vertical gaps of three horizontal slits (from horizontal projection).</td>
</tr>
<tr>
<td>$Inner(x_i, c_i)$</td>
<td>QDF function for geometric vector $p^w_i$ of $x_i$ and character class $c_i$. The $p^w_i$ comprises two elements; the first element represents the length from the top to the center line of the text line and the second element represents that from the bottom to the line shown in Fig. 5.</td>
</tr>
</tbody>
</table>

Fig. 5 Some geometric features.
The on-line character classifier employs Markov random field (MRF) models. We normalize an input character pattern linearly to a standard size while retaining the horizontal-vertical ratio and extract feature points from each stroke so that the start and end points are selected and the most distant point from the straight line between adjacent feature points is selected while the distance is greater than a threshold [28], as shown in Fig. 7 (b). Then, we match the feature points of each model with those of an input pattern with a DTW algorithm. We employ MRF to represent the character models and evaluate the similarities between the input pattern and the character models [29].

However, the off-line character classifier employs the modified quadratic discriminant function (MQDF) [30]. We apply pseudo-2D bi-moment normalization (P2DBMN) [31] to an input pattern, apply Gaussian blurring, and extract eight-directional features in the 8 x 8 regions shown in Fig. 7 (c). To improve Gaussianity, we apply Box-Cox transformation to each feature [32]. Then, we employ Fisher linear discriminant analysis (FLDA) [33] to reduce the 512-dimensional features to 160 features. MQDF measures the similarities between the input pattern and the prototypes. A combined classifier of the two above is formed by a sum rule in which the total score of a combined classifier is derived by adding two classifiers. Their weighting parameters have been optimized by the minimum classification error (MCE) criterion [34].

### 2.2 Estimation of Parameters

The main objective in estimating parameters is to tune the weighting parameters (λ) in the Eq. (1) to increase the scores of correct retrievals and decrease those incorrect retrievals. The conditional log likelihood (CLL) is widely used for estimating parameters in pattern classification [35].

The weighting parameters in (CLL) training are estimated on a dataset of training samples \( D = \{X^i, C^i\}_{i=1}^N \), where \( N \) is the number of input keywords. \( X^i = \{X_{1}^i, \ldots, X_{T}^i, X_{T+1}^i, \ldots, X_{S}^i\} \) denotes a sequence of retrieval results in a candidate lattice (\( X_{1}^i, \ldots, X_{T}^i \) are correct, and \( X_{T+1}^i, \ldots, X_{S}^i \) are incorrect), and the conditional probability for correct retrieval is formulated as Eq. (2).

\[
P(X^i|C^i) = \frac{\exp[\sum_{j=1}^{T} f(X_{j}^i, C^i)]}{\sum_{j=1}^{S} \exp f(X_{j}^i, C^i)} \tag{2}
\]

We introduce a threshold \( T_s \) to prune incorrect retrievals, which is determined so that correct retrievals produce scores higher than \( T_s \) while incorrect retrievals produce scores less than \( T_s \). To do so, the loss function is defined as the summation of the negative log likelihoods:

\[
L(\lambda) = - \sum_{i=1}^{N} \log P(X^i|C^i)
\]

\[
= \sum_{i=1}^{N} \log[\exp(f(X_{j}^i, C^i)) - \sum_{j=1}^{N} \exp f(X_{j}^i, C^i)] \tag{3}
\]

The stochastic gradient descent algorithm is used to optimize the weighting parameters:

\[
\lambda(t+1) = \lambda(t) - \varepsilon(t) \Delta L(\lambda) \tag{4}
\]

where \( \varepsilon(t) \) is the learning rate at the \( t \)-th iteration and \( \Delta L(\lambda) \) denotes the partial derivative with respect to the parameters:
3. Feature-Matching-Based Approach

We compared the ONHCR-based approach we propose for digital ink searches with the approach of keyword searches based on feature matching [13]–[15] in this research. Figure 8 has the flow diagram for it. We will describe all components in Fig. 8 in what follows.

1. Pattern generation: A search keyword in query pattern generation is input either from a keyboard as character code or from a tablet as digital ink. The first approach is code-based and the second is ink-based. We have focused on the code-based approach in this paper where each character of the keyword is converted to a standard on-line or off-line pattern and their sequence is prepared for matching, as shown in Fig. 9.

2. Over-segmentation: We employ the same method described in Sect. 2 to the over-segmentation of the target digital ink. Figure 10 has an example of over-segmentation.

3. Feature extraction: The on-line features or off-line features described in Sect. 2 are used for feature matching.

4. Feature matching: The keyword pattern in feature matching is directly matched with digital ink by block shifting. Given a keyword of \( m \) characters, we extract candidate regions from the digital ink starting from the current (initially first) primitive segment up to \( n \) consecutive primitive segments, where we consider \( n \) as wide as \([m, 4^m]\) so that we will not miss the occurrence of the keyword pattern. The search algorithm computes a matching score between the keyword pattern and all candidate regions, and produces the location of the candidate pattern if its matching score is less than a certain threshold. Then, it shifts the target regions by one segment and repeats this process. We represent the keyword pattern as \( X \) and the candidate regions as \( X_k \) where \( k \) is an index of candidate regions \([k = 1, \ldots, l]\). \( t \) is the number of candidate regions and it is \( 4m - (m - 1) = 3m + 1 \).

Keyword pattern \( X \) for the approach based on on-line feature matching is a concatenated sequence of feature points of standard on-line patterns for constituent characters of the keyword. Each candidate region, on the other hand, is linearly normalized to the size of the keyword pattern shown in Fig. 11 and feature points are extracted from each candidate pattern. Then, \( X \) consisting of \( l_A \) feature points and \( X_k \) consisting of \( l_B \) feature points are represented as \( p_1, p_2 \ldots p_{l_A} \) for the former and \( q_1, q_2 \ldots q_{l_B} \) for the latter, where \( p_i (i <= l_A) \) and \( q_j (j <= l_B) \) are the feature point coordinates of \( X \) and \( X_k \). Then, the matching score between the keyword pattern and each candidate region is measured with the recurrence equation in Eq. (6), where \( d(i, j) \) is the Euclidean distance between \( p_i \) and \( q_j \).

\[
f(X, X_k) = DTW(i, j) = \min \left( \begin{array}{c} DTW(i - 1, j) \\ DTW(i - 1, j - 1) \\ DTW(i, j - 1) \end{array} \right) + d(i, j)
\]

Unlike the on-line approach above, keyword pattern \( X \) in the approach based on off-line feature matching is a split sequence of standard off-line features of constituent characters for the keyword. The keyword pattern is matched with all paths that belong to a candidate region by using a tree matching strategy and the highest score of these is output. Figure 12 outlines the tree matching strategy. Each path has \( m \) candidate patterns where each candidate pattern is composed of a primitive segment or multiple consecutive segments. The off-line feature values are the same.
### Table 2 Cumulative character recognition rates in Kondate.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Top rec. rate</th>
<th>5th cumulative</th>
<th>10th cumulative</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-line</td>
<td>78.30</td>
<td>95.11</td>
<td>97.95</td>
</tr>
<tr>
<td>Off-line</td>
<td>79.99</td>
<td>96.86</td>
<td>98.46</td>
</tr>
<tr>
<td>Combined</td>
<td>80.06</td>
<td>96.86</td>
<td>98.50</td>
</tr>
</tbody>
</table>

as those described in Sect. 2.1 and they are extracted from each candidate pattern. Then, $X$ consisting of $m$ character patterns is represented as $v_1, v_2, \ldots, v_m$, while $X_k = \{X_i^k\}$ denotes a set of paths in a candidate region, where $i$ is the index of the path and each $X_i^k$ consists of $m$ candidate pattern and is represented as $w_1, w_2, \ldots, w_m$. Here, $v_i$ and $w_i$ are $d$-dimensionality feature vectors denoted as $v_i = [v_{i1}, v_{i2}, \ldots, v_{id}]$ and $w_i = [w_{i1}, w_{i2}, \ldots, w_{id}]$, where $d$ is the dimension of feature vectors. Then, the matching score is calculated with Eq. (7), where $\| \|$ is the Euclidean metric.

$$f(X, X_i^k) = \sum_{i=1}^{m} \|v_i - w_i\|$$ (7)

Some output results may overlap in the feature matching-based approach (both on-line and off-line) and the one with the lower score are pruned in the same way as that with the ONHCR-based approach.

### 4. Experiments and Evaluation

#### 4.1 Sample Pattern Databases

To evaluate how well the proposed keyword search approach performed, we employed the “Hands-Kondate” on-line handwritten text database (in brief, Kondate). Kondate is written by 100 participants and it included 13,680 lines of text. Table 2 summarizes the character recognition performance using three character classifiers introduced in Sect. 2.1: On-line, Off-line, Combined.

We carry out 5-fold cross validation. The 13,680 text lines were split into five blocks, consisting of 2,736 text lines each. Four blocks are used for training, and those remaining are used for testing. However, as the approach of keyword searches based on feature matching does not need to be trained we also used test data for the ONHCR-based approach to evaluate how effectively the approach based on feature matching performed. The keyword set for testing included 48 keywords composed of two characters, 81 composed of three characters, 51 composed of four characters, 25 composed of five characters, and 13 composed of six characters, which in total corresponded to 4,495, 6,315, 3,728, 1,798, and 842 times in the test data for digital ink.

We use the “Hands-Nakayosi” database (Nakayosi for short) [36] for training the character classifiers and geometric scoring functions. Nakayosi is a set of on-line handwritten character patterns written by 163 participants with each contributing 11,962 character patterns.

We generate keyword patterns of individual characters from the Hands-Tehon database, which included 7,722 character patterns with a correct number of strokes that were in order (Kanji: 7116, Kana: 169, Roman characters: 52, Numerals and Symbols: 385).

We evaluated how efficiently the search methods performed by using the $f$-measure

$$f\text{-measure} = \frac{1}{\frac{1}{r} + \frac{1}{p}}$$ (8)

$$r = \frac{\text{Number of correct search}}{\text{Number of search keywords in target data}}$$ (9)

$$p = \frac{\text{Number of correct search}}{\text{Number of searched items (include noise)}}$$ (10)

where $r$ is the recall defined in Eq. (9) and $p$ is the precision defined in Eq. (10). The recall rate was used to measure the tolerance to search errors, while the precision rate was used to measure the tolerance to search noise. The $f$-measure is an indicator of the overall performance of the search system.

#### 4.2 Results and Discussion

We evaluate the proposed approach to digital ink searches in a series of experiments. The first experiment is used to investigate how the different recognizers affected performance and how the geometric context contributed to it. We test the on-line, off-line, and combined recognizers by employing and not employing the geometric context.

Tables 3 and 4 summarize the performance of the three classifiers with and without the geometric context. We can see:

- The geometric context contributed to improving the $f$-measure. By combining geometric models, the averaged $f$-measure of the recognition-based on-line method is improved from .841 to .876; the averaged $f$-measure of the recognition-based off-line method is improved from .850 to .872; the averaged $f$-measure of the recognition-based combined method is improved from .913 to .932.
The combined recognizer always produced the best score for the f-measure in comparison with the on-line or off-line recognizers alone. Its superiority to the other two for the f-measure was more significant than that for the character-recognition rate listed in Table 2.

- **The longer the keyword, the higher the f-measure,** probably because of the larger amount of information employed for the search.

Because we have normalized each feature function $f_i (i = 1 \cdots 6)$ using variance normalization, we can consider that the larger the estimated weight $\lambda_i$ is, the more important the feature function $f_i$ is. Therefore, Table 6 implies that the most important feature function is the character classifier feature function $f_6$, then the next important feature functions are the inner gap feature function $f_3$ and the segmentation feature function $f_5$.

The second experiment compares the ONHCR-based approach with the one that is based on feature matching, which employed the same features as the former. Table 5 summarizes how well the latter performed.

By comparing Tables 3 and 5, we can see that the ONHCR-based approach outperforms the feature-matching-based approach both in on-line recognition v.s. on-line features and off-line recognition v.s. off-line features.

This is probably because the ONHCR-based approach reflects deformation models in terms of discriminant functions such as MRF and MQDF, while the feature-matching-based approach cannot exploit prior knowledge about pattern matching and can only employ geometrical or shape measures of similarity.

Although on-line and off-line recognizers have been combined in the approach based on ONHCR and produced the best performance, on-line and off-line methods of feature matching have not been integrated in the approach based on feature matching. This is because this integration would not be trivial and it remains another interesting topic for future research. If the two methods based on feature matching were optimally integrated, however, their union (correct if at least one method) would represent ideal recall and their intersection (correct if both methods are correct) would represent ideal precision. Table 5 suggests that optimal integration might produce better performance than the combined method that is ONHCR-based, but this seems extremely difficult from the low precision of their union and
poor recall of their intersection.

Although we excluded the ink-based feature-matching based approaches in this paper, ink-based methods may perform better in searches for one’s own handwriting.

5. Conclusion

We propose a character-recognition based approach to keyword searches on on-line handwritten Japanese text and compared it with approaches based on feature matching.

The character-recognition based approach integrated the scores of character classifiers and geometric context using discriminative training. The geometric context contributed to improving the f-measure.

Whether on-line features or off-line features are employed, the approach based on character recognition is superior to those based on feature matching probably because the character-recognition-based approach reflects deformation models in terms of discriminant functions, while the feature-matching-based approaches cannot exploit prior knowledge about pattern matching and only employ geometrical or shape measures of similarity.

The optimal integration of on-line and off-line features might produce better performance than the character-recognition-based approach from investigations into the unions and intersections of on-line and off-line methods of feature matching, but this seems extremely difficult from the low precision of the unions and poor recall of the intersections.

Of the character-recognition based approaches, the combined recognizer always produced the best scores for the f-measure in comparison with the on-line or off-line recognizers alone. Its superiority to the other two for the f-measure was more significant than that for the character-recognition rate.

However, compared to the ONHCR-based approach which requires a large number of training patterns, the feature-matching-based approach requires a limited number of labeled patterns for a given character-class. This makes the feature-matching-based approach extremely easy to extend to new languages.

Both approaches have a common characteristic in that the longer the keyword, the higher the f-measure.

Acknowledgements

We would like to thank Prof. Cheng-Lin Liu and Dr. Xiang-Doing Zhou for the valuable discussions we had with them and the sound advice they gave us.

References


