Chapter 1

Evaluation

1.1 Evaluation of Other HWR Systems

The recognition rates reported in the literature for other HWR systems are shown in figure 1.1 borrowed from **?** (**?**). As a general trend it can be noted that the recognition rate of most systems lies between 85% and 95%. **?** (**?**) believe that it is possible to achieve a recognition rate of up to 98% for regular scripts. On fluent scripts, however, they regard it as difficult to achieve a recognition rate above 90%. The systems marked with an asterisk in figure 1.1 perform recognition for Chinese or Japanese characters. The other systems recognise Latin script, hence, they are not an optimal comparison measure for the HWR engine developed in this thesis. The recognition rates of the Chinese and Japanese systems lie below 90%. That performance measure sets a general context in which the prototype system developed in this work is evaluated. However, the results of other systems are only comparable to the overall recognition accuracy of the present system. Due to the different technical background and the aim of the system it is difficult to draw conclusions from the sole comparisons of numbers. (See section 1.3.2.2 for more information.)

1.2 E-Learning Module Evaluation

Generally, there are two main directions in the evaluation methodology for e-learning applications: the educationalist's approach and the software developer's approach. Therefore, the evaluation of an e-learning system is a complex task and requires optimisation work on the account of both the conceptual designer of an e-learning application as well as the software developer.

The e-learning part of the prototype is a sample module that is used to exemplify one usage scenario of the HWR engine. It mainly shows the plausibility of the approach. A detailed analysis of the e-learning application using ISO9126 (**?**) is not applicable, since the e-learning part of the software has not been optimised in any way. The focus of the thesis is not to implement a full e-learning application, but to create an analytical handwriting recognition engine. It might be a prospect for future work[1] to optimise the e-learning part and build a fully-developed e-learning application for Japanese characters, but that would be outside the scope of this thesis. For these reasons, there will only be an evaluation of the HWR module.

1.3 Evaluation of the HWR Engine

1.3.1 General Considerations for Evaluation

The performance of a recognition system can be measured in terms of speed, accuracy and memory requirement. While statistical systems offer high speed, but have large requirements on memory, structural methods have lower speed, but have a smaller memory footprint (**?**). The system developed and evaluated in this thesis is a structural system. It can be expected that the system has a relatively low performance speed, but moderate memory requirements compared to other systems, since the system is not just a structural, but an analytical recognition system.

The factors memory requirements and speed are not of great interest in the context of this system. The system is an on-line system, but it exclusively performs single character recognition. The focus lies a lot more on a detailed analysis of one character, rather than the high-speed recognition of a stream of characters. The system is an interactive system. The recognition of a single character is an in-depth analysis of the structure of that character and returns a profound feedback to the user. Especially in an interactive learning context, the

[1] See section **??**

Source	Method	#category	Style	#learning	#test	Rec. rate
Liu'91 [75]	struct	6,763	flu-regular	N/A	N/A	90%
Kawamura'92 [45]	statis	2,965	careful/free	380 PC	20 PC	94.51/91.78%
Lin'93 [65]	struct	5,400	regular	1 PC	10 PC	87.4%
Liu'93 [76]	struct	13,000	flu-regular	N/A	N/A	93%
Hamanaka'93 [28]	statis	1,064	regular	54,028	52,944	95.1%
Chou'94 [16]	struct	5,401	regular	3 PC	17 PC	94.88%
Wakahara'95 [133]	struct	2,980	careful/free	120 PC	36 PC	97.6/94.1%
Lay'96 [60]	struct	5,401	regular	N/A	5 PC	96.35%
Kim'96 [47]	struct	1,800	free	4 PC	6 PC	93.13%
*Nakagawa'96 [88]	struct	3,345	fluent	N/A	11951 PW×30	80–90%
Chou'96 [18]	struct	5,401	flu-regular	5 PC	15 PC	93.4%
Wakahara'97 [134]	struct	2,980	careful/free	120 PC	36 PC	98.4/96.0%
Kim'97 [48]	HMM	1,800	free	4 PC	6 PC	90.3%
Zheng'97 [149]	FARG	3,755	regular	N/A	6 PC	98.8%
Xiao'97 [139]	struct	3,755	flu-regular	35 PC	3 PC	93.9
Nambu'98 [94]	struct	3,942	flu-regular	200 PC	200 PC	89.7%
Kuroda'99 [58]	statis	1,000	regular	25 PC	10 PC	94.34%
*Okamato'99 [99]	statis	3,345	fluent	11951 PW×40	11951 PW×41	86.32%
*Yasuda'99 [143]	HMM	3,057	fluent	10038 PW×100	10038 PW×20	85.89%
*Tanaka'99 [119]	combined	3,356	fluent	Nakayosi	Kuchibue	87.6%
Zheng'99 [150]	sta-struct	3,755	mixed	100 PC	7 PC	95.52%
*Akiyama'00 [1]	struct	3,345	fluent	6690 PW×150	11951 PW×3	88.58%
Shin'02 [113]	struct	2,965	regular	90 PC	24 PC	99.28%
*Kitadai'02 [53]	struct	3,345	fluent	9309 PW×163	11951 PW×120	87.2%
Tokuno'02 [121]	HMM	1,016	fluent	50,986	42,718	92.0%
Velek'02 [129]	combined	3,036	fluent	3,669,089	54,927	94.14%
Nakai'02 [92]	HMM	1,016	fluent	34 PC	34 PC	93.1%
Matic'02 [79]	neural	4,400	N/A	80 PC	20 PC	97.3%
Rowley'02 [106]	struct	6,847	natural	5 million	85,655	94.45%

PC: per category
PW: per writer
*Tested on TUAT Kuchibue database

Figure 1.1: Recognition rates reported in the literature

user is supposed to work with the system's feedback. Recognition speed is interesting for evaluation if the user enters a stream of characters. In that case recognition speed can be expressed as a factor that relates recognition speed with input speed. **?** (**?**) state that on-line recognition systems need only be fast enough to keep up with the writing. Further, they report average writing rates of 1.5-2.5 characters/s for English alphanumerics and 0.2-2.5 characters/s for Chinese characters. In a system that performs single character recognition the user has the impression of instant recognition. The recognition speed of a single character is below 1 second, thus below average writing speed for Kanji characters.

Memory requirements are negligible in this system for a similar reason. The recognition of one character does not require much memory compared to the recognition of a stream of characters. Additionally, the main system engine does not run on a small mobile computing device with low memory capacity, but runs as a service on a standard PC. With the advanced memory capacities of today memory is not an issue. Nevertheless, even mobile devices are equipped with enough memory to enable the system to perform analytical character recognition.

For the reasons given above, the evaluation of the analytical handwriting recognition engine will be limited to different types of accuracy measurements. The memory footprint of the e-learning and recognition system is around 35MB, including an overhead the .NET framework. It is difficult to measure the exact memory consumption without the .NET overhead. In a minimised state where the UI does not need to be shown the memory footprint drops to 1,200KB. The memory footprint of the recogniser lies around 25MB, including the .NET overhead, the estimated consumption without is 800KB.

1.3.2 Development of Appropriate Evaluation Metrics

1.3.2.1 Choice of Evaluation Subjects

It is difficult to perform an accuracy evaluation of a recognition system that can be compared to other systems. The methods the systems use in order to perform their recognition are diverse. There is always a trade-off between robustness, performance and accuracy. The prototype that is subject to this evaluation performs a different task than most of the other recognition systems. It analyses the characters not only for the purpose of recognition, but attempts to create feedback on how well the input matched the character model in structural terms. That means concretely, the system can analyse an input with an expected result and can perform the same for an unknown input by assuming the best match as the expected result. The analysis yields an output that includes more information than pure pattern matching. It rather includes structural linguistic information.

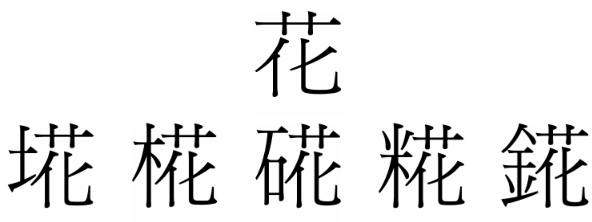

Figure 1.2: Similar characters that can be confusing to learners: Five different Kanji share the same radical in the *tsukuri* position (on the right).

For example, the characters 埖, 椛, 硴, 糀 and 錵 all share the substructure on the right: 花. Figure 1.2 shows the substructure on top and below five characters that are using the structure. The system analyses and distinguishes substructures. If the system is set up to recognise an input with an expected result the output contains a confidence value about the input quality concerning that character. Additionally, the error recognition module returns information about the substructures that were found in the input. The similarity of the characters from figure 1.2 is known to the recognition system because of the identical substructures. If the system identifies the input as a character with identical parts with respect to the expected character the output will contain the information as a substructure confusion error type.

This detailed analysis creates a unique requirement for evaluation. Not only recognition accuracy must be measured and compared to other systems, but a new evaluation for the recognition of substructures is needed. It seems reasonable to measure the recognition accuracy in a plain percentage of correctly recognised characters. Precision and recall are the correct measurements for the error recognition, where characters with incorrect substructures are identified. That way, true positives, false positives, true negatives and false negatives can be distinguished. Since the system can recognise pure substructures as well, it would be interesting to see the accuracy of a radical recognition.

The lexicon is a sample lexicon that contains 50 characters. That circumstance has to be kept in mind when comparing the evaluation results to other systems. Creating a lexicon entry for a single character is a laborious task. It would be outside the scope of the thesis to create a large lexicon. Nevertheless, the characters have been chosen by graphic and semantic criteria in order to create more possibilities for confusion. There are many characters using the same key radical, other characters that simply look similar, which might evoke difficulties for both the learner and the system. Three experiments will be conducted in order to evaluate the analytical handwriting recognition engine:

1. **Overall recognition accuracy**
2. **Substructure recognition accuracy**
3. **Error recognition accuracy**

The metrics details for these experiments will be presented in the following sections.

1.3.2.2 Overall Recognition Accuracy

The overall recognition accuracy (experiment 1) will be calculated as a measure of the n-best matches for a character input. N is defined as the lowest position in the list of n-best matches that will still be considered as a *match* of a character. That is, if N is set to 1, then only the best match will be considered. If $N = 3$, the best three matches will be taken into account. The overall recognition accuracy A_N is defined here as the weighted percentage of correctly recognised characters when taking the N best matches into account. The number of samples will be called S. r is an integer that marks the position in the list of n-best matches. The values for r that will be considered for the evaluation lie on the interval $[1, N]$. If an input produces a confidence value for a character recognition high enough to be in a list position $r \leq N$, then $\frac{1}{r}$ will be added to the number of correctly recognised characters. That means, if a character is the best match in the list of n-best matches, then it is ranked on top, therefore $r = 1$. Hence, $\frac{1}{r} = \frac{1}{1} = 1$ will be added to the number of correct matches. If the correct character is the second best match, $\frac{1}{2}$ will be added and so forth. If the input has been correctly recognised as the r-best match, but $r > N$ or, alternatively, the input has not been recognised at all, nothing will be added. f is a helper function that returns a value according to the relation between r and N. The variable r_i denotes the position of the i^{th} input character in the list of n-best matches for that input sequence and character. r_i serves as an input value for f. A_N will be calculated as follows:

$$f(r, N) := \begin{cases} \frac{1}{r} & \text{if } r \leq N \\ 0 & \text{if } r > N \end{cases}$$

$$A_N := \frac{1}{S} \cdot \sum_{i=1}^{S} f(r_i, N)$$

An example for the application of that term would be to set $N = 1$. Only the best match will be considered for an input. If 100 input sequences are tested, then $S = 100$. If, for example, the k^{th} sample of the S_i yields a high confidence value for the character the sequence is supposed to represent and the character becomes the most salient in the list of best matches and occupies the first position. Then $r_k = 1$ and $f(r_k, N) = f(1, 1)$ yields 1. Therefore, in the summation of all the $f(r_i, N)$ the value 1 will be added for each correctly recognised character. Say, 69 of the 100 input characters were correctly recognised, then $A_N = A_1 = \frac{1}{100} \cdot 69$. That would equal a result of 69%. The key to this evaluation method is of course the value of N. Multiple experiments with different N-values will be conducted for best comparability.

Baseline. Different possibilities exist for the definition of a baseline for that experiment. A baseline based on equal distribution would be $\frac{1}{C}$ with C representing the number of characters in the database. For $C = 50$ the baseline would then be $\frac{1}{C} = \frac{1}{50} = 0.02$. That is already a very low baseline. The larger the lexicon grows the lower the baseline for evaluation. For a lexicon that contains around 2,000 characters and covers the Jōyō Kanji the baseline would come down to $\frac{1}{C} = \frac{1}{2000} = 0.0005$. This baseline is certainly not helpful to measure the quality of a recognition system. A natural baseline could be created by using a corpus study to determine the most frequent characters and always use the most frequent one as a result for the system's recognition. The flaw of that idea is that the database of the system does not include all characters yet, so it would again not be very helpful to use a baseline like that. The test set of characters for both the prototype system here and the baseline system would have to be a handwritten text, so that the baseline system could yield a good result within it's technical limits.

Since there is no method to create a baseline for comparability of this system, I am defining an *idealised fictitious baseline* (IFB) value based on the formula for the computation of A_N including generous assumptions about the accuracy of the recognition of an IFB system that always hits:

Assume, $N = 3$ and all the sample characters are among the first three matches in the list of matches, equally partitioned. Then one third would yield $r = 1$ with $f(r, N) = f(1, 3) = 1$, another third would yield $r = 2$ with $f(r, N) = f(2, 3) = \frac{1}{2}$, and the last third would yield $r = 3$ with $f(r, N) = f(3, 3) = \frac{1}{3}$. That leads to the calculation of an IFB value (B):

$$
\begin{aligned}
B &:= \tfrac{1}{S} \cdot \Big(\sum_{j=1}^{\frac{S}{3}} f(1,3) + \sum_{j=1}^{\frac{S}{3}} f(2,3) + \sum_{j=1}^{\frac{S}{3}} f(3,3)\Big) \\
&= \tfrac{1}{S} \cdot \Big(\sum_{j=1}^{\frac{S}{3}} 1 + \sum_{j=1}^{\frac{S}{3}} \tfrac{1}{2} + \sum_{j=1}^{\frac{S}{3}} \tfrac{1}{3}\Big) \\
&= \tfrac{1}{S}\Big(\tfrac{S}{3} + \tfrac{S}{3 \cdot 2} + \tfrac{S}{3 \cdot 3}\Big) = \tfrac{1}{3} + \tfrac{1}{6} + \tfrac{1}{9} = \tfrac{11}{18} = 0.61
\end{aligned}
$$

The defined baseline value of 0.61 will be used for the overall recognition accuracy experiment. However, it is important to note that the IFB value is not a natural baseline in the sense of what the performance of a most simple system would be like.

1.3.2.3 Substructure Recognition Accuracy

The substructure recognition accuracy (experiment 2) will be measured in a similar way as the overall recognition accuracy. In fact the same metrics will be used. See the previous section for details on the metrics. The accuracy A_N will be measured as a weighted percentage value of correct recognition results, where the recognition as second-best or third-best (generally n-best, with $n > 1$) will be devaluated. This reduction of value is expressed mathematically by using the multiplicative inverse r^{-1} of the rank r in the list of n-best matches as a summand for the weighted sum of correct recognitions. The equations for calculating the accuracy have been given in section 1.3.2.2.

The main difference is that many substructures occur in more than one character. In order to account for that, a substructure recognition will be regarded as *correct*, if the input yields the intended substructure from one of the characters that contain the substructure. That fairly generous interpretation of what is correct considers the fact that the substructures should be recognised for any of the characters. In a future version of the database the substructures should be stored only once and then be shared among all characters that contain it. The IFB value for the substructure recognition will be 0.61, the same number and using the same line of argument from section 1.3.2.2.

1.3.2.4 Error Recognition Accuracy

The error recognition accuracy (experiment 3) is more difficult to evaluate than the pure recognition accuracy. The question arises, what kind of errors are expected to be found and what kind of error corrections should be provided. Error recognition here means that the system compares a known character with a given input stroke sequence. The input can contain flaws that do not represent the character appropriately. The prototype is expected to find the flaw and propose a correction. The evaluation method will be a classical *precision and recall* method. There are two possible cases in the input stroke sequence:

- There is an error in the input with respect to the desired character.
- There is no error in the input.

There are two possibilities for the system to react:

- The system identifies an error in the input with respect to the desired character and gives feedback on that error.
- The system does not identify an error in the input with respect to the desired character.

Those possibilities yield a 2×2-matrix: Table 1.1 gives an overview about the possible results combined with the reality of the sample.

	Error in input	No error in input
Found error (TP+FP)	true positive (TP)	false positive (FP)
No error found (FN+TN)	false negative (FN)	true negative (TN)

Table 1.1: Possible results as a basis for precision and recall analysis.

The precision P and recall R of the error recognition are computed with the standard formulae:

$P = tp \cdot (tp + fp)^{-1}$
$R = tp \cdot (tp + fn)^{-1}$

Additionally, the *F-measure* will be used, however only for an equally important interpretation of precision and recall. Therefore, $\beta = 1$ will be fixed:

$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R}$
$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$

1.3.3 Experimental Results

Having presented the experimental metrics in the previous sections, the experimental results and the conclusions drawn will be presented in this section.

1.3.3.1 Overall Recognition Accuracy Results

For the experiment concerning the overall recognition accuracy, two test writers wrote all 50 characters from the database as an input sequence for the system. That process yielded 100 input stroke sequences. The input was stored in order for the experiment to be repeated without physically re-entering the characters. Five sample characters from each writer haven been removed from the test set for detailed analysis. With five characters from each writer in the development set, there were 90 characters for the test set. The test set characters have been used only for testing, not for detailed analyses. The development set characters have been tested as well. The recognition process and the recognition results have been examined in order to improve the recognition quality.

The final evaluative run of the test set has been repeated 10 times, with different values for N. All of the integers from 1 to 10 have been used. The *weighted number of recognised characters* (WNRC) changes according to the number of characters recognised in each rank in the list of n-best matches. Table 1.2 shows the results in greater detail. With each increment of N, that is with decreasing strictness the recognition results rise. The range of the weighted accuracy results goes from 0.70 for $N = 1$ up to 0.81 for $N = 5$. Taking into account even greater ranks up to $N = 10$ in the list of n-best matches increased the result slightly to 0.82. This development can also be seen in the chart in figure 1.3.

N	A_N	WNRC/90	1	2	3	4	5	6	7	8	9	10
1	0.70	63.00	63	-	-	-	-	-	-	-	-	-
2	0.74	67.00	63	8	-	-	-	-	-	-	-	-
3	0.77	69.33	63	8	7	-	-	-	-	-	-	-
4	0.80	71.83	63	8	7	10	-	-	-	-	-	-
5	0.81	72.63	63	8	7	10	4	-	-	-	-	-
6	0.81	72.80	63	8	7	10	4	1	-	-	-	-
7	0.81	73.23	63	8	7	10	4	1	3	-	-	-
8	0.82	73.35	63	8	7	10	4	1	3	1	-	-
9	0.82	73.58	63	8	7	10	4	1	3	1	2	-
10	0.82	73.68	63	8	7	10	4	1	3	1	2	1

Table 1.2: Overall recognition results.

Result Interpretation. As mentioned earlier, the characters bear a lot of potential for confusion, since many of the database characters have been chosen by their key radical or by similar graphical appearance to other characters already in the database. That special choice of characters seems to explain why there is such an increase in accuracy when more ranks are taken into account. The similar characters will be confused by the system and appear in the lower ranks (higher r-values) in the list of n-best matches. Even though all the results, even the one with $N = 1$ are considerably above the IFB value of 0.61. Generally, the accuracy results are considerably lower than those of other systems (see section 1.1). However, it has to be noted that the aim of this system is not to provide a handwriting recognition highly optimised for speed and accuracy. Hence, there is little comparability for these results. The prototype of the analytical handwriting recognition fulfils an additional task that the other systems do not aim at. That obviously leads to a trade-off between recognition performance and accuracy of the analysis. The aim is to create a deep analysis of a character structure and be able to provide feedback on the input. These abilities of the system will be evaluated in the following sections.

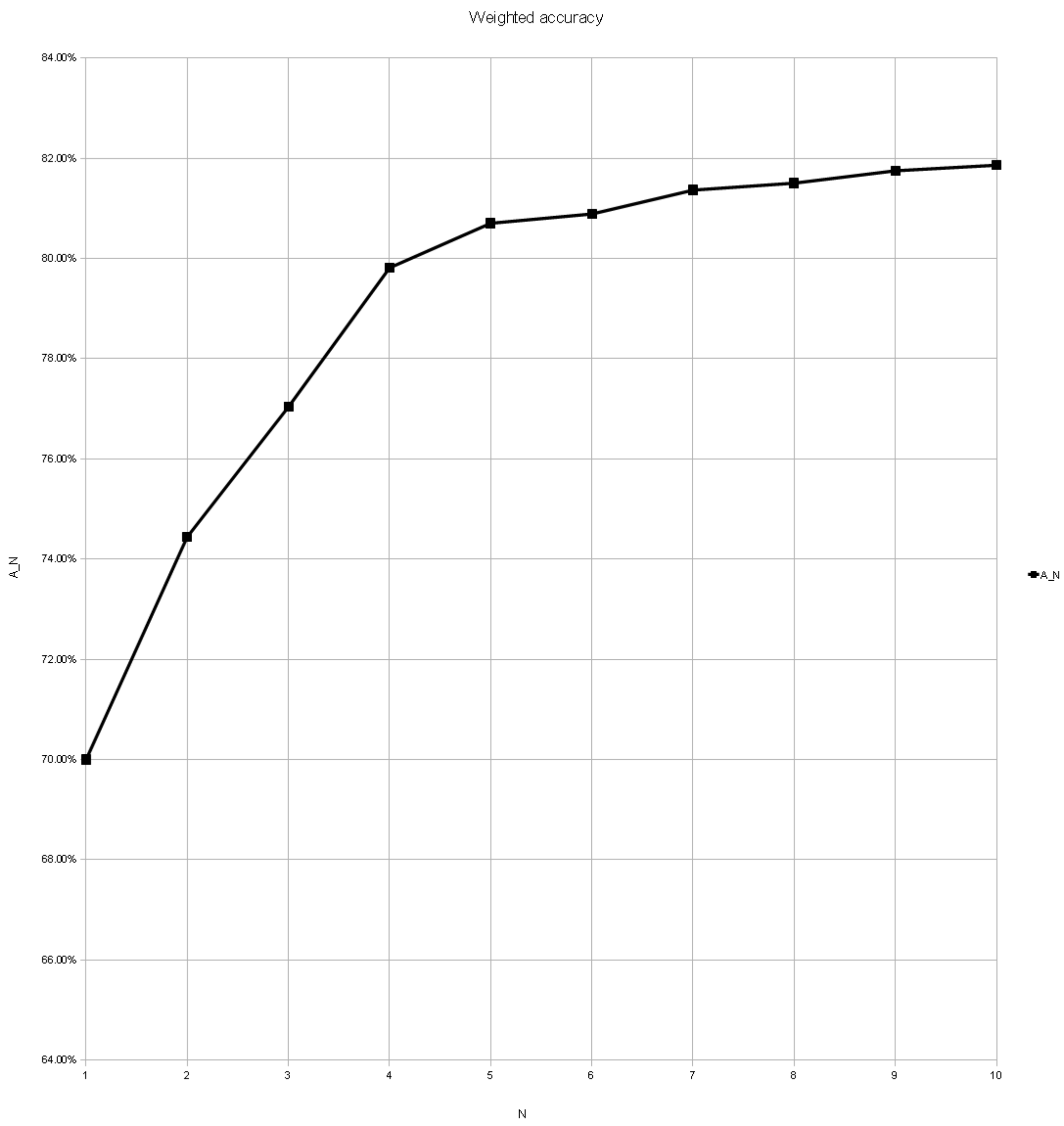

Figure 1.3: Overall recognition results chart

The analysis of recognition errors among the development set characters revealed that there are two main types of errors:

1. **Different stroke order**
2. **Similar characters**
3. **Segmentation ambiguity**

The type 1 and type 2 errors probably occur for all handwriting systems. But type 1 errors are problematic especially for overlaid or continuous handwriting recognition. Those types of error can be resolved by using a statistical model like a *Hidden Markov Model.* The type 3 error, segmentation ambiguity is a problem that occurs mainly in overlaid handwriting recognition for full characters. In analytical handwriting recognition the problem occurs for substructures. The beginning and the end of a substructure can only be found by completely analysing the substructure, the previous and the following substructure. That process has shown to be time-consuming.

1.3.3.2 Substructure Recognition Accuracy Results

For the evaluation of the substructure recognition it would have been desirable to use the same input data with which the overall evaluation had been conducted. That was not possible because it would have required to analyse the input data and split it into substructures manually. The input data for the evaluation of the substructures has been provided by three writers each of whom entered 25 substructures, i.e. a total of 75 radicals and Graphemes. That means 75 different tokens, not 75 different types. Each type has been written once by each writer. This input set is a pure test set it has not been split or used for development work. The results of the substructure recognition evaluation are more comparable to the results of other systems. The metrics for the substructure recognition are essentially the same as for the overall evaluation. The difference is that only the first three entries in the n-best list will be considered. The structures are smaller, need less analysis, do not have further substructures except the individual strokes. As expected, the evaluation results are higher than those of the overall recognition accuracy. The general trend can be seen in figure 1.4, the complete data is laid out in table 1.3.

N	A_N	WNRC/75	1	2	3
1	0.85	64	64	-	-
2	0.89	67	64	6	-
3	0.90	67.33	64	6	1

Table 1.3: Substructure recognition results.

Result Interpretation. As expected, the accuracy of the recognition results for the substructures was higher than the overall character recognition accuracy. The substructure recognition is more robust as it does not have a segmentation problem. Each input sequence is known to be a full substructure. During the recognition process the database needs to be searched for substructures. If a matching structure is found the process is completed. The same pattern like in overall recognition accuracy can be observed concerning the lower positions in the list of n-best matches.

1.3.3.3 Error Recognition Accuracy Results

The error recognition evaluation required more manual work than the overall recognition or substructure recognition evaluation. This is due to the fact that erroneous characters had to be produced. Therefore, this evaluation type has to be seen more as a qualitative analysis, rather than a pure quantitative evaluation.

Test Data Creation. One writer created 30 characters, including 20 characters with errors, i.e. 20 characters where one radical had been replaced with another one or with garbled input. Figure 1.5 shows the characters 没 (Jap. pron. *botsu*; Eng. 'to sink') and 海 (Jap. pron. *umi*; Eng. 'sea') both in print and handwritten versions. The two characters share the same key radical. It is variant of the 'water'-radical 水 on the left side (marked in red). The two radicals on the lower right side are fairly distinct and therefore no cause of confusion. In fact 海 (umi/'sea') contains the 'mother'-radical 母 (marked in purple), while 没 (botsu/'to sink') contains the 'again'-radical 又 (marked in purple), which looks very similar to the 'father'-radical 父 (not contained in the figure). This mother/father-analogy has nothing to do with the true meaning of the characters, it can just serve as a mnemonic for a learner. This fact could be a cause for semantic confusion due to the mnemonic, but

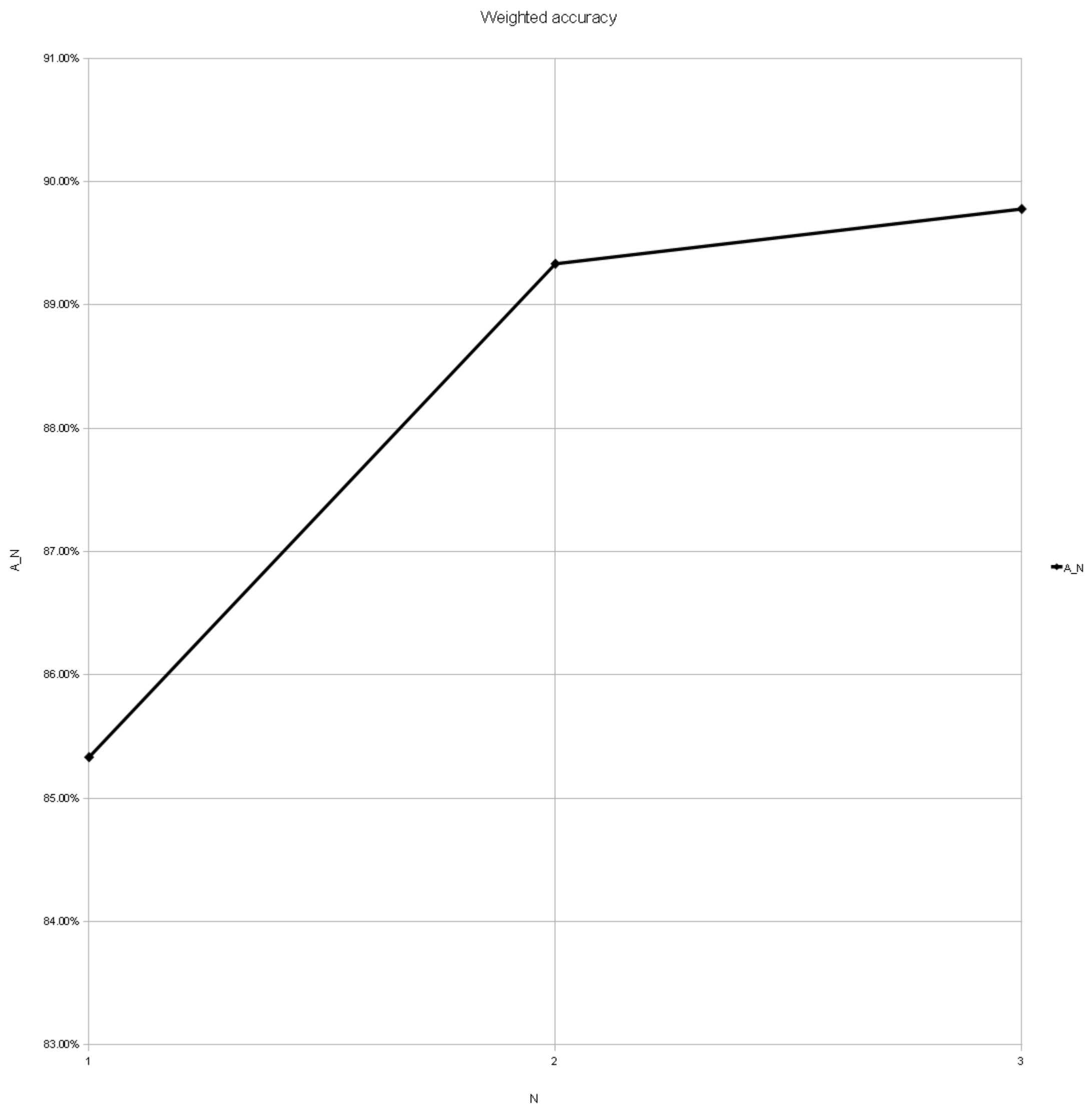

Figure 1.4: Substructure recognition results chart

clearly the two radicals 'mother' 母 and 'again' 又 offer little room for a graphical confusion between them. The upper radical on the right in both characters is a small two-stroke radical that can easily be confused graphically with another small two-stroke radical (marked in green). This danger of confusion exists both for a human and for the character recogniser. Non-existent blended characters like the one shown in figure 1.6 have been created as erroneous input for the error recognition evaluation test set. In case of a confusion of the two radicals in the upper right corner of the character the user might produce an input like the one in figure 1.6.

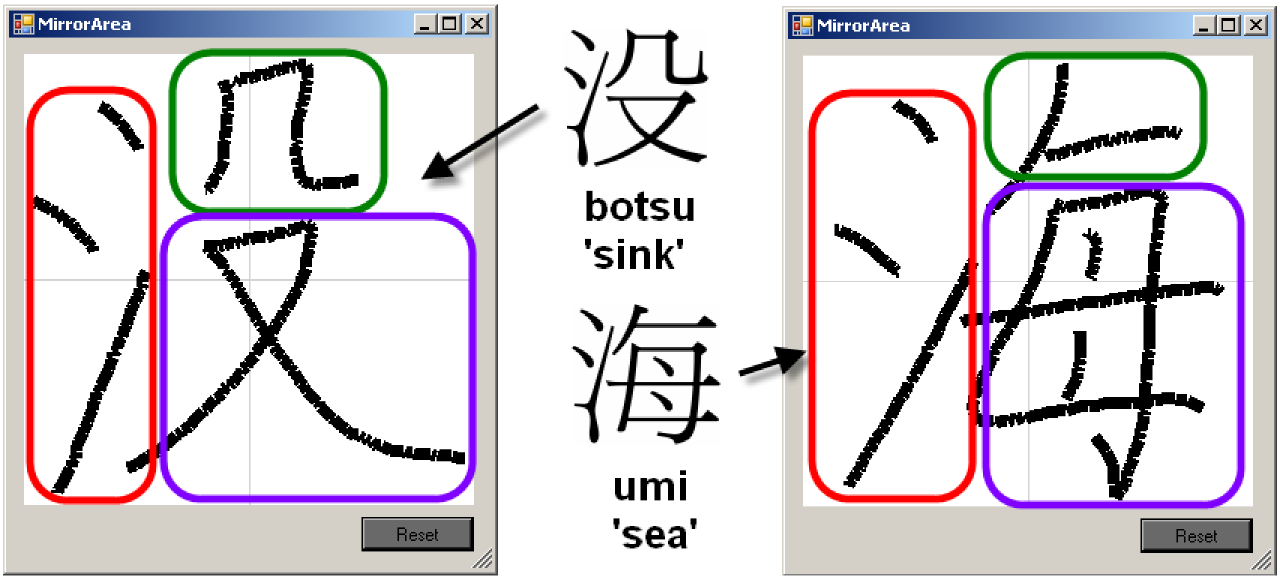

Figure 1.5: Characters 没 (*botsu*, 'sink') and 海 (*umi*, 'sea')

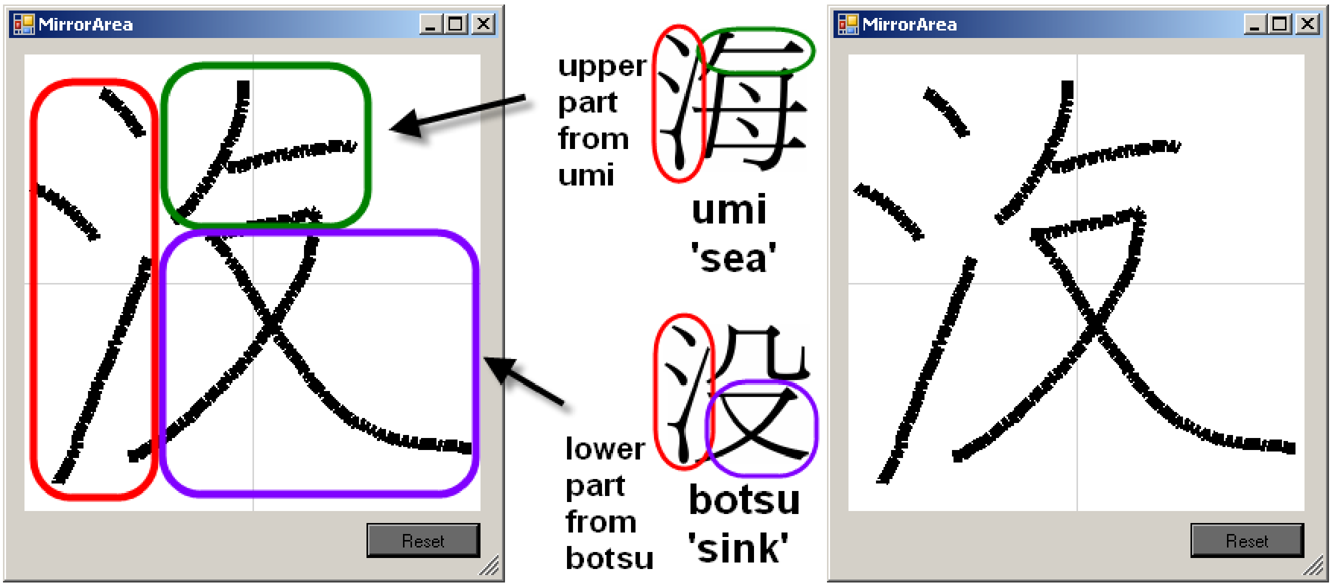

Figure 1.6: Confusion between the characters 没 (*botsu*, 'sink') and 海 (*umi*, 'sea')

Error Recognition Result Data Set. Table 1.4 shows the results of the error recognition test run. In section 1.3.2.4 it has been argued that the correct evaluation method for the error recognition was a precision and recall measurement. The problem with a pure precision and recall analysis is that the results in table 1.4 are not 2-way, but 3-way. The error recognition engine returned the three result sets *character contains error*, *characters does not contain error* and *character could not be recognised*. Now the question arises whether the unrecognised characters should be regarded as incorrect or correct. If there was an error in the input character and the character could not be recognised, this could be regarded as a *true positive* because implicitly the system states that an error was found. Arguably, if no recognition is possible that means that there is an error

Total of 30 chars (20/10)	Error in input	No error in input
16 found error (TP+FP)	14 (TP)	2 (FP)
9 no error found (FN+TN)	6 (FN)	3 (TN)
5 unrecognised (?)	3 (?)	2 (?)

Table 1.4: Error recognition results as a basis for precision and recall analysis.

in the character, therefore an error has been found. However, that is just an interpretation of the results and would increase the error recognition performance undeservedly. On the other hand, if the input was a correct character that could be not recognised, then assuming the undefined result to be an error message would add to the number of *false positives*, lowering the performance result of the error recognition. The error recognition is a sub-component of the character recognition engine, it cannot work independently from it. Hence, either way it would be unfair to include the unrecognised characters into the evaluation of the error recognition.

In order to ensure a fair evaluation, but also to provide a result set qualified for a 2-way evaluation, the unrecognised characters are exempt from the evaluation of the error recognition. Table 1.5 shows the data set that was used for the calculation of precision and recall without the unrecognised characters. The total input set consisted of 30 characters, 20 of which contained errors, i.e. 10 correct characters. After exempting the unrecognised characters from the evaluation, there is a total of 25 characters, 17 with errors, 8 correct characters.

Total of 25 chars (17/8)	Error in input	No error in input
16 found error (TP+FP)	14 (TP)	2 (FP)
9 no error found (FN+TN)	6 (FN)	3 (TN)

Table 1.5: Error recognition results excluding the unrecognised characters as a basis for a 2-way precision and recall analysis.

Result Interpretation. The precision P and recall R can be computed from the values given in table 1.5. Precision:

$$P = tp \cdot (tp + fp)^{-1}$$
$$= 14 \cdot (14 + 2)^{-1}$$
$$= 0.875$$

Recall:

$$R = tp \cdot (tp + fn)^{-1}$$
$$= 14 \cdot (14 + 6)^{-1}$$
$$= 0.7$$

Since there are no comparable evaluation results, the *F-measure* will be used only for an interpretation of precision and recall. Therefore, $\beta = 1$ will be fixed:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$
$$= 2 \cdot \frac{0.875 \cdot 0.7}{0.875 + 0.7}$$
$$= 0.78$$

The results are difficult to interpret because there are no other systems for comparison that perform a similar analytical handwriting recognition. Precision is slightly below 90%, meaning if an error had been recognised, it had been recognised correctly most of the times. That is valuable for the sample application, the e-learning environment. The number of falsely detected errors is low. The recall is not impressively high, but well above the obvious 50% baseline for a 2-way precision and recall calculation, that would result from an equally distributed assignment of *error* or *no error* to all characters in the test set.

1.4 Conclusions

In this chapter, the evaluation of the handwriting recognition engine has been presented. The overall recognition has an accuracy of 70%-82%, depending on what ranks ($r \leq N$) is considered among the n-best matches. That is below highly optimised recognition system and shows the trade-off that had to be taken in order to perform a detailed internal analysis of the characters. The recognition of substructures has better results, accuracy is around 85%-90%, depending on N. That value is more comparable to other recognition systems that do not perform a detailed analysis of substructures. The error recognition showed a satisfying result of 87.5% precision at a recall rate of 70%. The next chapter of the thesis draws overall conclusions from the work.

Document created on Sunday 27$^{\text{th}}$ June, 2010 at 13:26