How Much Handwritten Text Is Needed for Text-Independent Writer Verification and Identification*

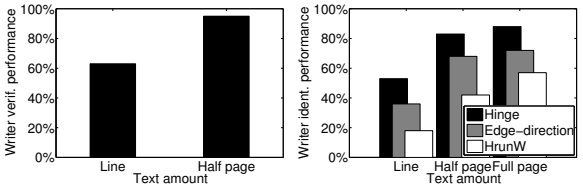
Axel Brink Marius Bulacu Lambert Schomaker
Dept. of Artificial Intelligence, University of Groningen
P.O. Box 407, 9700 AK Groningen, The Netherlands
a.a.brink@ai.rug.nl, bulacu@ai.rug.nl, schomaker@ai.rug.nl

Abstract

The performance of off-line text-independent writer verification and identification increases when the documents contain more text. This relation was examined by repeatedly conducting writer verification and identification performance tests while gradually increasing the amount of text on the pages. The experiment was performed on the datasets Firemaker and IAM using four different features. It was also determined what the influence of an unequal amount of text in the documents is. For the best features, it appears that the minimum amount of needed text is about 100 characters.

1. Introduction

The reliability of writer verification and writer identification depends on the amount of text in the handwritten documents: the more text is present, the more evidence is available. This is relevant for the forensic application domain, where the amount of text varies from snippets to complete letters. It is also particularly relevant for automatic texture-based methods such as Hinge, Fraglets, run-lengths, direction histograms etc. [7] Such methods build a model of writer specific features that can only be accurate when the number of measurements is large, thus they need a sufficient amount of text. This has been shown in previous studies: In [5], a writer verification experiment yielded a performance of 63% using a single line of text per page, while the performance on half a page was 95%. In [4], a writer identification experiment yielded 53%, 83% and 88% on a single line, half a page and a full page, respectively, using the best feature. It showed a similar pattern for two other features. These scores are illustrated in Figure 1.



(a) Writer verification using a combination of *micro* and *macro* features [5].

(b) Writer identification using three different features [4].

Figure 1. Previous studies show that classification performance increases when the amount of text increases.

The objective of this study is twofold. The first objective is to gain insight in the relation between text amount and the performance of current writer verification and identification systems into more detail. This facilitates judging the reliability of the outcome of automatic classification based on the amount of text. The second objective is to derive a rule of thumb for the *minimum* amount of text that is required for reliable classification using the best features. Knowing such a minimum is useful when designing new datasets for automatic writer verification and identification. Such a minimum cannot be firm, because performance does not only depend on the amount of text but also on other factors such as feature quality, text quality and database size.

2. Method

The relation between text amount and performance was examined by repeatedly conducting writer verification and identification performance tests while gradually increasing the amount of text on the pages. The amount of text on the pages was controlled by *wiping* text from complete pages of existing datasets. It was

*This paper was published in ICPR 2008. Available online: http://figment.cse.usf.edu/~sfefilat/data/papers/WeBT6.2.pdf

also determined what happens when one of the compared documents contains little text (a partially wiped page) while the other contains a lot (a full page). In the following subsections, the building blocks of the experiment are presented. The last subsection shows how these blocks were put together in a series of performance experiments.

2.1. Datasets

Two handwriting datasets were used: Firemaker [10] and IAM [6], which can be obtained from the respective authors. Firemaker is a dataset consisting of Dutch text handwritten by 251 students; each student wrote four pages of which we used page 1 and page 4. Page 1 contains a fixed text consisting of 612 alphanumeric characters; page 4 contains free text of 252 characters on average. Both pages were written in natural handwriting and scanned at 300 dpi. The IAM dataset contains handwritten English text written by 657 subjects; the number of pages written by the subjects varies. The pages were written in natural handwriting and scanned at 300 dpi.

Both datasets were split in two parts: part *Q* represents questioned documents and part *S* represents sample documents. This mimics the realistic forensic scenario where the authorship of questioned documents is to be determined while a database of documents of known authorship exists. Not all documents in the original datasets were put in *Q* or *S*: pages with less that 200 characters were discarded and the same happened to pages of writers that wrote only one page. Of the remaining pages, for every writer, the page containing the most text was added to *S* and the next page was added to *Q*. Additional pages were discarded as well. For the Firemaker dataset, the resulting sets *Q* and *S* both contained 192 scans (118 pages were discarded). For the IAM dataset, *Q* and *S* both contained 298 scans (378 pages were discarded).

2.2. Text wiping

We present $\texttt{line\_wipe}_n$, a method to wipe handwritten text on a line-by-line basis until an estimated n characters remain. It is natural to use a line-based approach because it is straightforward and it keeps the appearance of the handwriting almost intact. The general idea is to wipe some lines completely and one line partially. This approach requires that the text lines can be separated well. This is not feasible in general free text but it is relatively easy in the Firemaker and IAM dataset, because the text lines are quite horizontal and the overlap between descenders and ascenders of consecutive lines is minimal. The locations of the text lines can be found with projection histograms. For the Firemaker and IAM datasets, this has already been done in [2] and [6] respectively. From these publications, the coordinates of boxes around the text lines were re-used for this experiment.

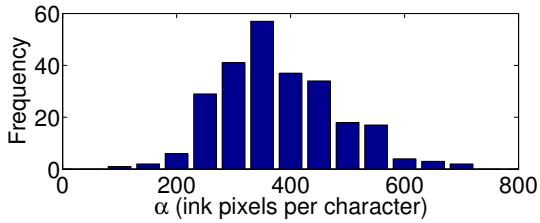


Figure 2. α varies on the scans of the Firemaker dataset.

Based on these boxes and a desired number of characters n, the handwritten text can be wiped. Ideally, after wiping, exactly n characters would remain. In practice, this is not possible, because it is not yet possible to segment free text into characters. Instead, n is converted to a desired number of ink pixels: $d = \alpha \cdot n$. The variable α denotes the average number of ink pixels per character on the page, which can be computed by dividing the number of dark pixels in the image by the number of characters in the transcription. As an example, Figure 2 illustrates the distribution of α on the scans of the Firemaker dataset.

The next step is to remove full text lines by filling the bounding boxes with white pixels. The lines are wiped in random order while maintaining a variable r, the number of remaining ink pixels on the page. When wiping the next line would decrease r below d, the line contains too much ink to be removed entirely. That line is wiped partly as follows. First, the vertical projection histogram of the line is created and then smoothed using a Gaussian window. The valley that is closest to the point where $r = d$ is designated as the cutting point. The right part of the line is wiped starting from this point. This approach makes it likely that the cut is made between characters and not through characters, thus lowering the risk of damaging the letter shapes. See Figure 3 for an example.

2.3. Feature extraction

Four methods were used to compute writer features: *Hinge*, *Fraglets*, *Brush* and *HrunW*. These features will be briefly introduced; refer to the respective papers for the details. The *Hinge* feature [3] is a normalized histogram of angle combinations that are measured on the boundaries of the ink. This encodes slant and curva-

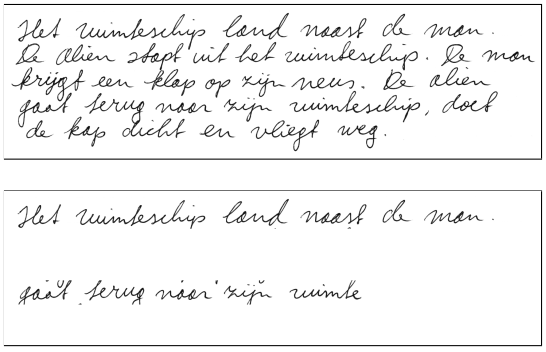


Figure 3. Text before (top) and after (bottom) $\mathrm{line\_wipe}_{n=50}$ was applied. In this example, actually 51 letters remained.

ture. The *Fraglets* feature (also known as fCO3) [8] is a normalized histogram of usage of ink blobs from a codebook. This encodes allograph usage. It uses a codebook that was trained on all four pages of the first 100 subjects of the Firemaker dataset. The *Brush* feature [9] is a histogram of ink intensities at stroke endings. It encodes pen landing and lifting habits. The *HrunW* feature [1, 9] is a histogram of horizontal white run-lengths. This encodes within- and between-letter spacing.

2.4. Writer verification

Writer verification is the process of determining whether two documents have been written by the same person. In forensic practice, the two compared documents are usually a questioned document and a sample document. In automatic classification, the documents are assigned to the same writer if (and only if) their feature similarities are below a threshold. This threshold is learned from the data: smooth distributions of between- and within-writer feature distances are created using Parzen windowing. Using these distributions, the expected ratio of false positives and false negatives can be balanced according to the preference of the user. Since such a preference was not available, the threshold was set such that the expected rate of false positives and false negatives was equal; the *equal-error rate* (EER). The EER was used as the performance measure for writer verification.

2.5. Writer identification

Writer identification is the process of searching for suspected writers in a database of handwritten sample documents, given a questioned document. The resulting hit list is sorted by feature distance to the questioned document and typically has a fixed size of 1 (*top-1*) or 10 (*top-10*). For performance assessment, every $q \in Q$ was once treated as the query. A performance measure for writer identification is the fraction of query documents for which a document of the same writer actually appears in the hitlist.

2.6. Experiment

In the experiment, the building blocks described in the previous subsections were put together to create graphs that show the relation between text amount and performance: while varying the number of characters n, the performance of both writer verification and writer identification were computed. This was done in two variants: in one variant, the documents in part Q and S of the dataset were both wiped such that they contained the same number of characters n. In the other variant, only the pages in Q were wiped, while the documents in S remained intact. This distinction respects the difference in text amount between the compared documents in forensic practice. Algorithm 1 outlines the experiment in pseudocode. The functions named `split_dataset`, `line_wipe`, `f`, `WV`, and `WI`, were described in the previous subsections (without explicitly naming them). The algorithm was run on both the Firemaker and IAM dataset, using the features Hinge, Fraglets, Brush, and HrunW.

Algorithm 1 Complete experiment for a dataset D and a feature function `f`. Compute performance of writer verification and identification, with equal and unequal amounts of text.

$Q, S = \texttt{split\_dataset}(D)$
for all n **in** $[4, 9, 16, 25, ..., 196, 200]$ **do**
  $QW = \texttt{line\_wipe}_n(Q)$
  $F_{QW} = \texttt{f}(QW)$ *(compute features)*
  $SW = \texttt{line\_wipe}_n(S)$ *(for equal text amount*)*
  $F_{SW} = \texttt{f}(SW)$ *(compute features)*
  $\mathrm{verif\_perf}(n) = \texttt{WV}(F_{QW}, F_{SW})$
  $\mathrm{ident\_perf}(n) = \texttt{WI}(F_{QW}, F_{SW})$
end for
* For unequal text amount: $SW = S$

3. Results

Figure 4 shows the results of the experiment. The eight graphs each represent a combination of a feature and a dataset. In each graph, the Top-1 and Top-10 performance and the verification EER are shown; each in a variant of equal-text amount and unequal-text amount. The graphs confirm that the Hinge is a strong feature, closely followed by the Fraglets feature. They also clearly show that increasing the amount of text also

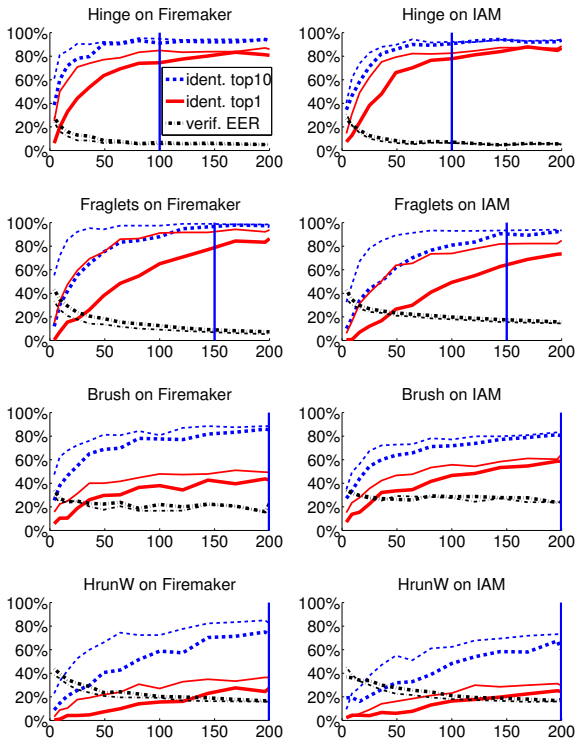


Figure 4. Relation between text amount (in characters; horizontal axis) and performance of writer verification and identification. Thick lines indicate equal text amount; thin lines indicate unequal text amounts. The vertical lines indicate a suitable minimum text amount.

increases writer verification and identification performance. From these graphs, a rule of thumb for the minimum text amount was derived by visual inspection. Although it is possible to do this using an exact method, that would not be appropriate for the inexact situation. As a rule of thumb, the following minimum amounts of text can be considered:

Feature	Min. text amount (chars)
Hinge	100
Fraglets	150
Brush	200
HrunW	200

These minima were drawn into the graphs as vertical lines. The graphs also consistently show a better performance for comparison of texts that have an unequal amount of text. Thus, it is always better to increase the amount of text in the database documents, even when the amount of text in the query documents remains the same.

4. Conclusion

As a rule of thumb, a document of 100 characters contains a good minimum text amount for text-independent writer verification and identification when using a strong feature such as Hinge. Any text above this minimum does not significantly increase performance any further; the feature itself becomes the limiting factor. For features that are less powerful, such as Brush, a reasonable minimum is 200 characters. These numbers are not absolute because the performance also depends on other factors such as the size of the database. In general, the more difficult the classification task, the more text is needed. In any case, more text is always better, even when the amount of text is increased in only one of the documents in pairwise comparisons. A follow-up to this study should consider text-dependent methods, since those are more similar to the manual methods used in forensic practice, where text samples can be very small.

References

[1] B. Arazi. Handwriting identification by means of run-length measurements. *SMC*, 7:878–881, 1977.

[2] M. Bulacu and L. Schomaker. Writer style from oriented edge fragments. In *CAIP 2003*, LNCS, pages 460–469. Springer, 2003.

[3] M. Bulacu and L. Schomaker. Text-independent writer identification and verification using textural and allographic features. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 29(4):701–717, 2007.

[4] M. Bulacu, L. Schomaker, and L. Vuurpijl. Writer identification using edge-based directional features. In *ICDAR 2003*, pages 937–941, 2003.

[5] S. Kabra, H. Srinivasan, C. Huang, and S. Srihari. On computing the strength of evidence for writer verification. In *ICDAR*, pages 844–848, 2007.

[6] U. Marti and H. Bunke. A full english sentence database for off-line handwriting recognition. In *Proc. of the 5th ICDAR*, pages 705–708, 1999.

[7] L. Schomaker. Writer identification and verification. In N. Ratha and V. Govindaraju, editors, *Advances in Biometrics: Sensors, Systems and Algorithms*, pages 247–264. Springer-Verlag, 2007.

[8] L. Schomaker, M. Bulacu, and K. Franke. Automatic writer identification using fragmented connected-component contours. In F. Kimura and H. Fujisawa, editors, *9th IWFHR*, pages 185–190, Tokyo, Japan, October 2004.

[9] L. Schomaker, M. Bulacu, and M. van Erp. Sparse-parametric writer identification using heterogeneous feature groups. In *ICIP*, volume 1, pages 545–548, September 2003.

[10] L. Schomaker and L. Vuurpijl. Forensic writer identification: A benchmark data set and a comparison of two systems. Technical report, NICI, Nijmegen, 2000.