Writer identification in medieval and modern handwriting

Tara Gilliam

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

University of York
Department of Computer Science

September 2011

Abstract

Writer identification is the task of associating a handwriting sample with the identity of the correct writer. It can be used to confirm or refute the authenticity of a document, or to link together documents produced by the same writer. This problem has applications in several areas, including forensics and palaeography – the study of historical books and writings.

Rigorous manual writer identification requires the exhaustive comparison of character details, and is very time-consuming, making computer automation of all or part of this process attractive. Most research into automated writer identification has originated in forensic science, although more recently applications to historical texts are increasing. With mass digitisation of texts on the rise in libraries and collections, organising this new data is a growing problem.

However, different types of writing have different characteristics, and require different handling. This thesis focuses on how medieval English manuscripts from the 14th–15th centuries compare to the contemporary handwriting datasets used for much of the research and feature development in this area.

The work presented here is based on an in-depth application of the grapheme codebook approach to offline writer identification. It finds domain-specific considerations throughout the process, particularly in grapheme creation and comparison and in the influence of document sources on system accuracy. Additionally, over the course of the data analysis, methods are proposed for the visualisation of extracted features, for quantifying the impact of sample source on identification accuracy, and for a nearest-neighbour-based verification system.

Contents

List of Figures

Acknowledgements

First, I would like to thanks my supervisors Prof. John Clark and Prof. Richard Wilson, and my assessor Prof. Jim Austin, for their advice and support throughout my research. Thanks also to the EPSRC for financial support under a PhD studentship.

For the provision of the data for this work I would like to thank the AHRC-funded project "Identification of the Scribes Responsible for Copying Major Works of Middle English Literature", and Prof. Linne Mooney and Dr. Estelle Stubbs in particular for their time and expertise in arranging access and identifications.

Finally I would like to thank my family and all those who have supported, encouraged, and distracted me over the last four years: Heather, Mark, Louis, Nick, Charles, Eliza, and Lin for all our extremely sensible and completely research-focused discussions, to all those in the NSC ~~pizza~~ research-student group, and to the members of YCCSA for widening my view of my research.

Author's declaration

I declare that all work in this thesis is my own, except where attributed to another author. Some of the results in this thesis have been previously published as follows:

Tara Gilliam, Richard C. Wilson and John A. Clark
Scribe Identification in Medieval English Manuscripts
20th International Conference on Pattern Recognition, 2010

Tara Gilliam, Richard C. Wilson and John A. Clark
Segmentation and Normalisation in Grapheme Codebooks
11th International Conference on Document Analysis and Recognition, 2011

Chapter 1

Introduction

Palaeography is the study of historical documents and writings. The majority of the work lies in transcription, interpretation, and dating of manuscripts, but a significant component is the identification of a manuscript's writer, or scribe[1].

Identifying the writer of a copy can have significant implications in confirming its authenticity. For instance, manuscripts associated with Geoffrey Chaucer's personal scribe, Adam Pinkhurst (Mooney, 2006), are likely to have been overseen by the author personally, allowing historians a degree of confidence in their fidelity. Scribal identification can also form an element of manuscript dating or locating, as well as indicating the extent of transmission of a particular text.

Scribal identification is a time consuming manual process, requiring expertise in the personal handwriting styles of up to hundreds of scribes working in a given time period. Minute details of the style of individual characters and pen strokes are examined and compared between manuscripts to confirm whether they were written by the same scribe. This process is usually complicated by the difficulty of bringing valuable manuscripts from different collections together for comparison (Davis, 2007).

The practicalities of this situation are in contrast to the other major contem-

[1] In historical works, the terms writer or scribe are distinct from author, as the composer of a text was frequently a different person to the scribe that produced any single copy.

porary application of writer identification: forensic or questioned document examination (FDE/QDE), carried out to provide evidence in criminal investigations or civil legal cases. Here experts typically have easy access to the documents in question, and a growing access to computerised writer identification systems and databases to aid them (Tapiador and Sigüenza, 2004; Srihari and Leedham, 2003; Bulacu and Schomaker, 2005b; Franke et al., 2003; Niels et al.). Although there is a substantial body of work available in writer identification, it largely originates in forensic science and little of it has been systematically targeted at historical data. This is significant as the approach and purpose of the writing can be very different to that of modern day script. This influences both the writers and their technique, and consequently the writing styles they produce.

Previous work has divided the influences on personal handwriting style into the *genetic* and the *memetic* (Schomaker and Bulacu, 2004), that is, innate and learned factors. With widespread literacy, modern-day writing is largely for personal use. Prior to the invention of the printing press, the most common role of a writer was essentially that of a copyist, producing texts to a predetermined specification. These disparate purposes swing the balance of personal and purely taught influences in opposite directions: the far higher degree of style enforcement in Medieval and earlier times is reflected in more standardised scripts. Texts were produced chiefly for public consumption on a medium (vellum, or other parchments) which was expensive, whereas the chiefly transient, private use of contemporary writing (and the typical lack of instruction beyond primary education) allows an individual style more freedom to develop.

Given this divide in the writing characteristics of each period, this thesis tests the hypothesis that **medieval and modern writing will have significantly different responses to writer identification techniques**. From this motivation the thesis is structured as an in-depth analysis of two datasets: a typical contemporary dataset widely used in writing analysis research, and a new medieval English manuscript dataset. The experiments carried out highlight the areas of writer iden-

tification in which medieval data requires special handling, as well as the methods which are robust to the dataset type. This thesis will therefore develop and evaluate techniques originating in modern writer identification for the domain of medieval English manuscripts. The following sections will look in more detail at the medieval writing process, before moving onto the computational aspect: setting the offline writer identification field in the context of its development as a biometric classification problem, and outlining the specific method on which the work in this thesis is based.

1.1 Medieval writing process

In the medieval period prior to the invention of the printing press, texts were produced by hand-copying a new or existing text onto parchment, using a quill pen and ink most likely manufactured by the writer creating the work. At a time when literacy (and writing in particular) was not widespread, these were typically professional scribes producing a wide-variety of texts such as legal documents, religious and secular literature, and collections of medical or scientific knowledge.

Scribes were usually involved in most aspects of book production. The writing material was usually parchment (also known as vellum, although this term sometimes refers particularly to the higher-quality parchments), made from cleaning, treating, washing, and stretching animal hides (usually calf, sheep, or goat). These were then folded and cut into booklets of pages known as quires, which were bound together to form books. The inner 'flesh' side and outer 'hair' side of the resulting parchment are often distinguishable, and can form part of the page texture and colour in image reproductions. Paper, generally imported, was also beginning to gain ground as a writing medium at this time.

Depending on the type of work and the size of the page, the text could be laid out across the full page, or in two or more columns. Scribes sometimes marked guides on their pages before writing by a combination of drawing margin lines,

pricking evenly down the page where each line should start, or drawing or scoring horizontal lines. Some texts were highly ornate, with miniature illustrations placed both in the margin and between the main text, gold or silver illumination, coloured inks, decorated or illustrated capitals, and decorated borders (marginalia). These may have been produced by the scribe who copied the work.

Quills were usually made from goose feathers. The shaft was cured to harden it, before the tip was trimmed and shaped into a nib. They required periodic resharpening, which can often be detected in a manuscript by the effect the changed nib-shape has on the ink trace. Unlike modern pens, writing with a quill required almost no pressure on the page, eliminating pressure variation from the possible writer-identifying features (Stokes, 2009).

The writing process itself was slow. Scribes sometimes noted the dates they started and completed a particular manuscript or work, and examination of these suggests that they typically averaged 4–6 sides per day, or 24–40 sides per week (Gillespie and Wakelin, 2011, p. 35). The speed of composition would be affected by the font used for the work: the more formal fonts such as Textura were associated with prestigious and higher-quality texts. They required the pen to be lifted after each stroke, while the faster cursive Secretary script was more typical in everyday documents. Font choice was, in turn, affected by both the content of the text and to some extent the language in which it was written, with Textura associated more strongly with Latin and liturgical works, and the more cursive scripts associated with vernacular English (Greetham, 1994).

Professional scribes would have been able to produce a range of fonts appropriate to the text to be copied, significantly complicating the identification task. In addition to this multiple scribes often worked on a single manuscript, and modified their personal style to match each other with the aim of making writer transitions smooth and undetectable. From a writer identification viewpoint, this style imitation is essentially a form of forgery, executed for the purpose of professional presentation of a manuscript.

From this process, we can see that medieval texts have very different writing characteristics to modern texts, notably the text samples generated for typical writer identification work. These tend to be written freehand in an individual's undisguised personal style, with cleanly separated lines of text, a single source language, similar-size samples, and often an identical pen-type. The IAM dataset shows most of these characteristics and is representative of a typical modern writer-identification dataset. The medieval dataset, sourced from a wide range of manuscript collections, has none of these features and is illustrative of the medieval text production described above. Section 3.2 gives further details of the datasets used in this work.

As the majority of writer-identification research to date has taken place on contemporary datasets, these period-specific differences suggest that there will be aspects of current state-of-the-art techniques that are suboptimal or ineffective on medieval data. The next sections describe the general process of writer identification in its context as a biometric pattern recognition problem.

1.2 Writer identification in context

Handwriting is an example of a biometric, a personal attribute that can be used to identify an individual. The theoretical basis for this comes from the field of classification or pattern recognition, which is concerned with attributing naturally varying measurements to the correct identity. This section will give a brief overview of the field of biometrics and the pattern recognition process it applies, and conclude with a summary of the most closely related research areas.

1.2.1 Writer identification as a biometric

"Biometrics is the science of recognising the identity of a person based on the physical or behavioural attributes of the individual..." (Jain et al., 2008, Preface). Physical attributes are measurable biological qualities that a person possesses,

such as fingerprint, iris, or face recognition. Behavioural attributes are distinctive traits in the way a person performs some action, e.g. gait, speech, or handwriting. The main current uses of biometrics are in secure authentication (also known as verification) of a person's identity, e.g. fingerprint scanning in the US-VISIT programme[2], and iris recognition[3]. The longstanding and widespread use of personal signatures to authenticate transactions also falls into this category.

Handwriting has been recognised as a valid biometric, with the evidence of Forensic or Questioned Document Examiners (FDEs/QDEs) being judged legally admissible (Srihari et al., 2002; Davis, 2007; Jain, 2002). Its main uses are in expert witness regarding the authenticity of a particular text, and in determining the authorship of a historical text where this is unknown or in question.

In Jain et al. (2005), seven characteristics are listed by which to judge a potential biometric; Dunstone and Yager (2009) lists thirteen. Particularly where handwriting is concerned (Schomaker, 2007) these are primarily factors such as universality, uniqueness, permanence, and measurability, i.e. the biometric should be possessed by all members of the relevant population, it should be individually distinctive, it should remain distinctive, and it should be possible to collect the data in a suitable form. Further considerations include social, practical, and systems design issues. Jain et al. (2005) and Impedovo and Pirlo (2008) note that no biometric will meet all such criteria, but must be application-appropriate.

Handwriting generally meets the main criteria for a useful biometric. It is universal, as the population in practical use comprises those who have already produced written documents; its uniqueness has been demonstrated in e.g. Srihari et al. (2002); and its measurability gives rise to the study of feature extraction and selection that occupies much of the writer identification literature. The question of permanence is more interesting. It is well-known that there are many factors affecting writing style, both habitual (memetic) and physiological (genetic)

[2]http://www.dhs.gov/files/programs/gc_1208531081211.shtm
[3]http://www.ukba.homeoffice.gov.uk/travellingtotheuk/Enteringtheuk/usingiris/

(Schomaker and Bulacu, 2004). Aging, and associated physical conditions that may develop, clearly have some impact but have not been found to prevent identification (Walton, 1997).

The process of writer identification is carried out through implementing a biometric classification system. The writer-specific technique that has been chosen for this problem domain is the grapheme codebook process. Descriptions and models of these two systems are given in the following sections.

1.2.2 Biometric systems

The following sections provide an outline of the processes and operation of a biometric system. More detailed discussions of these (including error and accuracy measurements) can be found in e.g. Bolle et al. (2003); Dunstone and Yager (2009); Jain et al. (2008) or Jain et al. (2004).

System Processes

A biometric system must be able to collect, process, store, and compare biometric samples, and output a decision as to whether any two samples were collected from the same individual (Dunstone and Yager, 2009). These components support the processes of *enrollment*, and at least one of *identification* and *verification*.

Enrollment The first required phase of a biometric system is enrollment (also called registration), when the system is initialised with biometric samples from the verified individuals which it must be able to recognise. The system components required at this stage are collection, processing, and storage. Collection involves a physical sensor to accept biometric inputs; processing includes any automatic and manual processing of the input required to produce a good representation of the biometric; following which one or more verified biometric samples may be stored for the individual. These samples form the data on which the system will rely when making authentication decisions, and is assumed to be accurate.

After enrollment, the system is able to compare any incoming samples to its database of registered users and decide whether or not a match has been found. There are two main modes of operation: verification and identification.

Verification In verification mode (also called authentication), the system checks whether an incoming sample matches the identity that an individual claims. This requires first that the claimed identity is in the system database, and that the processed sample matches the system's record for that individual closely enough that it judges them to have been produced by the same person. The system may or may not add the newly collected sample to its database. The system components involved in this process are collection and processing (for the new sample), storage (for retrieval of existing samples and possible retention of the new), comparison (to determine how closely the new sample matches existing data for that individual), and decision (to determine whether the outcome of the comparison is close enough).

Identification In identification mode (also called recognition), the individual does not need to claim an identity: the new biometric sample is compared against all individuals known to the system. This requires a far greater number of comparisons than verification. Additionally, two comparison thresholds are required in the decision component: the first, as before, determines whether a sample is close enough to a specific individual to match, but the second must determine whether the sample is a close enough match to *any* individual registered on the system. The components involved in this process are again collection and processing (for the new sample), storage (for retrieval of all existing samples and possible retention of the new), comparison (to determine how closely the new sample matches existing data for all individuals), and decision (to determine whether the outcome of the comparison is close enough to match a single individual).

Measuring system accuracy

Unlike most authentication systems, which require an exact match between tokens (such as passwords), biometric tokens will naturally have some variability. This implies that its matching system must also tolerate variation. It also suggests that should a perfect match be found, it is likely to indicate that a copied sample has been used in an attempt to fool the system. Too high or too low a threshold will both produce authentication errors, and various measures have been used to describe how well a system achieves this goal. The accuracy of a biometric system is usually phrased in terms of its error rates.

Many biometric systems perform verification, for which there are two standard types of error: False Accept Rate (FAR), the chance of incorrectly attributing a presented sample to the wrong individual, and False Reject Rate (FRR), the chance of failing to match a presented sample to the correct individual. These are called Type I and II errors respectively, and are also known as False Match/Non-Match Rates (especially when considering identification rather than verification problems). Either rate can be trivially reduced to zero by rejecting or accepting all samples presented to it. They must therefore be used in conjunction to determine the trade-off between them, giving the error characteristics of a particular system. The information in this graph is frequently summarised using the Equal Error Rate (EER), the point on the curve where the FAR and FRR have the same values.

The same types of error are present in identification systems, but it is important to compare the error characteristics of the identification problem itself against the verification problem. Verification checks whether a single claimed identity matches the sample presented, whereas identification checks for a match with any known identity. If a discriminator has a particular error rate in verification, this chance of error will occur at every individual match considered in the identification problem. The corresponding identification error rate is therefore related to the product of the error on every single match. This demonstrates that identification is, statistically, a far harder problem than verification, and the accuracy require-

ments for features which perform the former task are much higher (Daugman, 2000).

1.2.3 Biometrics as Classification

The process of determining this identification as accurately as possible has been formalised in the field of pattern recognition, or classification. The basic stages in a pattern recognition task will be outlined in turn below, along with a discussion of their corresponding biometric system components and writer identification stages.

Preprocessing This stage involves the collection and preparation of the data into a suitable form to work with, and is obviously very specific to the type of data being classified. In the case of images, this step can comprise scanning, digitisation, and manual or automatic cleaning and enhancement. It is carried out in the biometric system components of collection and processing.

For offline writer identification, this typically includes the digitisation of the document images, skew correction, region-of-interest/text-block selection, conversion to black-and-white (binarisation), and any text-component extraction required to support later processing stages.

Feature Extraction There is typically far too much information in the input data to classify it directly, and much of this information will be (at best) irrelevant to determining the correct class. The pattern recognition process therefore relies on measuring or extracting elements of the original data that are useful indicators of the class to which it belongs. These elements are known as *features*, and their development forms a major part of each pattern recognition application domain.

In the domain of writer identification, the bulk of the research has been focused on this stage, and a detailed survey of proposed writer identification features is given in Section 2.3. In the biometric system, this stage is also included in the processing component.

Additionally, there are some optional steps that may be carried out in deriving improved features from those initially extracted. Features may be combined in various ways to generate new features (feature generation), or they may be put through a preliminary testing phase to pick out the most accurate predictors (feature selection). Both of these stages are more typical in the development of new features than the implementation of a stable biometric system, as the latter usually relies on techniques which have already been proven reliable.

Classification Once each input sample has been represented by a set of features, samples can be compared to determine which group, or class, or measurements they are most likely to be from. In writer identification, these classes correspond to individual writers: each person is modelled as a generator of a set of handwriting 'patterns' of their own personal style. The concrete instances of these patterns extracted from document images form the class of writing samples attributed to that person. The process of attributing a previously unknown sample is known as *classification*.

A wide array of algorithms have been developed to do this (many of these can be found in Bishop (2007)). Biometrics generally employs techniques from *supervised classification*, which requires accurate samples from known individuals to train the classifier, gathered in the Enrollment phase. These training set samples may be used to generate a template or prototype of writer's typical style, or they may be used directly to describe the range of patterns that a writer produces.

In classification terms, the set of features used to describe the samples is known as the *feature space*, and a single sample's list of measurements is its *feature vector*. Each feature vector represents a single point in the feature space, and the distance between two vectors illustrates how similar their originating samples are. This distance measurement occurs in the comparison component of a biometric system. A good set of features will therefore produce feature vectors which are close together when samples are from the same person, and far apart when

samples are from different people: ideally, the points from a person's samples would form a single cluster which does not overlap that of any other writer. In this case, the identity of the writer of an unknown document may be determined (the biometric system process of identification) by:

- measuring the document sample's values for each feature (feature measurement), e.g. average character body width or length of descending strokes
- plotting the resulting point in the feature space containing the training data
- determining which cluster (i.e. class) the point is closest to (classification)

This is illustrated in more detail in Section 3.4.3.

The corresponding process of verification is similar, but requires an additional piece of information: the writer (i.e. class) the unknown sample is expected to belong to. Instead of looking for the closest cluster, verification considers whether or not the sample is likely to belong to the expected writer's cluster of points. In both verification and identification, the decision component of the biometric system is responsible for this final stage of classification.

1.3 Grapheme Codebook Method

Although a wide variety of techniques are in use in writer identification, this work focuses on the grapheme codebook method (as described in Bulacu (2007), also known as the Fraglets feature) as a basis for the dataset analysis. It was chosen for its high identification performance, relatively low processing and metadata requirements, and similarity to manual writer identification processes. These elements were important in this work as the medieval dataset has no transcriptions available, and the methodological similarities provide a good reference point for those with a humanities background in the target application area. The final sections of this introduction give an outline of the grapheme codebook process.

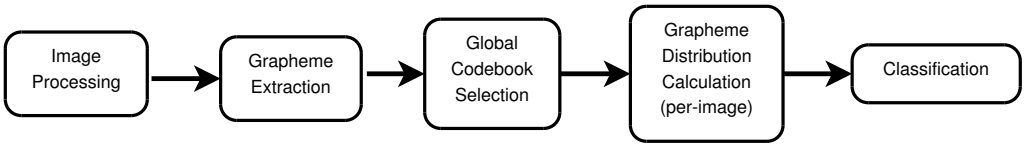


FIGURE 1.1: Outline of the grapheme codebook identification process.

The codebook method is based around the segmentation of text into **graphemes**. These are character-scale text fragments formed by dividing a cursive ink trace heuristically. Although the aim is to provide an approximately character-like segmentation of the text, the character content or meaning of the grapheme is not used, so whole, partial, or merged characters are equally valid. A good heuristic will give a consistent segmentation, i.e. given a similar ink trace, will produce similar output graphemes. This allows comparison of the distribution of ink trace shapes formed by a writer.

Preprocessing stages for the codebook method are outlined in Figure 1.1. They consist of binarising the input image, then extracting the connected-components (all joined sections of ink trace pixels). These are segmented into graphemes and scaled to a fixed size for comparison. Each image in the training and test sets is represented by its constituent graphemes, and the dataset as a whole is represented by the union of these. From this whole-dataset grapheme collection, a fixed number (typically 100–1000) are selected to form a global codebook: the reference basis by which the images in this dataset will be measured.

The next step is to measure the features and calculate the feature vector for each image in the dataset. Each grapheme in an image is compared against all the graphemes in the codebook, and tallied against the one it matches most closely. Once this has been dome for all graphemes in an image the tally is normalised to sum to 1, giving a probability distribution. This is the image's feature vector with respect to the selected codebook. Once all image feature vectors have been calculated, and optional feature selection or extraction step may take place. The final identification step is classification: comparing the feature vectors in the train-

ing set against those calculated from unattributed images to determine their most likely writer.

1.4 Thesis Layout

The thesis is therefore structured as an investigation of each stage of the grapheme codebook process. Chapter 2 reviews the writer identification literature, looking particularly at approaches to feature development. The pre-processing steps of image binarisation and cleaning, and connected-component extraction and basic segmentation, are developed and applied in Chapter 3. It also expands upon the choice of the grapheme codebook method for feature extraction, and describes a consistent experimental methodology that will be followed for the work in this thesis. Baseline identification results are provided for later comparisons.

The experiments in Chapter 4 test alternatives to the initial stages of grapheme segmentation, scaling, and grapheme image comparison. Chapter 5 analyses combinations of grapheme features, and proposes and evaluates codebook selection criteria. Chapter 6 considers the final stage of identification, comparing classification strategies and analysing both types of dataset against an example verification system. Finally, Chapter 7 summarises the conclusions drawn from the experiments in this thesis and identifies limitations and areas for further work. The main findings of this thesis are that:

- Grapheme aspect-ratio is a writer-specific feature in modern handwriting but not in scribal handwriting
- Translation-invariant grapheme comparison improves modern writer identification rates, but does not affect scribe identification
- The style of the document from which a sample originates strongly influences scribe identification rates, but has minimal effect on the modern dataset

Chapter 2

Literature Review

This chapter comprehensively reviews the available techniques in computational writer identification. Although the dataset for this work is drawn from a palaeographic background, the relevant techniques are from automated document processing and biometrics, and it is these areas that will be surveyed.

Writer identification is based on a behavioural biometric: a learned activity from which personally identifying data can be extracted. As such, the identification process falls into the theoretical framework of Pattern Recognition. As an area of document analysis, it is also linked to areas of handwriting research such as Optical Character Recognition (OCR).

Within automated writer identification, methods are divided by the type of information they employ: motion and pressure captured at the time of writing (online information), or image-based data only (offline information). Within offline identification, a further loose categorisation is made according to the level of writing-specific knowledge required as input. This ranges from none, where images are treated as pixel patterns or textures, up to detailed information on the character set and textual content of each sample.

This chapter outlines related areas of work before examining the existing literature in details, including the current state of automated writer identification for the historical domain.

2.1 Related Areas

The following sections summarise the areas of writing analysis research which are closely related to image-based writer identification: motion-based writer identification, signature verification, and optical character recognition.

Online Writer Identification Writer identification is split into two top-level branches: online and offline writer identification. Online identification analyses writing samples that have been captured during production, making use of the dynamic motion, direction, timing, and pressure information available (Li and Tan, 2009; Li et al., 2007), as well as shape-based data (Namboodiri and Gupta, 2006; Blankers et al., 2007), or a combination of approaches (Schlapbach and Bunke, 2007a). In contrast, offline writer identification uses only measurements drawn from the static image of the completed ink trace.

Online writer identification has a high identification accuracy, and the additional information available makes it generally superior in this regard to purely offline techniques. A review of the current state of the field is given in Chapran (2006), as well as an example identification system; as a large part of this field involves signature text in particular the references in the section below are also relevant.

However, online techniques are clearly only applicable where the writing samples to be studied can be recorded in progress. This excludes it from application to historical and archived documents, as well as most forensic applications.

Signature Verification The most developed branch of writing-based biometrics is signature verification: the task of determining whether a given signature was produced by the claimed identity. In particular, online signature verification benefits from a high degree of standardisation, with benchmark databases available (Guyon et al., 1994; Garcia-Salicetti et al., 2003; Ortega-Garcia et al., 2003; Ye-

ung et al., 2004) and several related ISO standards[1].

Current state-of-the-art systems tend to comprise a selection of motion and duration statistics (Tan et al., 2009; Chan et al., 2007; Li et al., 2007). Online information is usually preferred to static or purely shape-based information. Reviews of this area are given in Plamondon and Lorette (1989) and Leclerc and Plamondon (1994), with more recent work presented in Jain et al. (2008, Chapter 10) and Impedovo and Pirlo (2008).

Optical Character Recognition The aim of OCR is to convert text from a graphical form (either machine print or handwriting) into a machine-encoded text format. Although online handwriting OCR is well-established (Tappert et al., 1990), this summary will focus on offline handwriting OCR as the area of greatest overlap to writer identification.

OCR is the most widespread form of writing image processing, found in e.g. commercial and home scanning software packages, as well as more specialised applications such as postcode and address reading for mail sorting (Liu, 2003; Lee and Leedham, 2004) and automated bank cheque processing (Cheriet et al., 2007; Liu, 2003). Thorough reviews of standard methods and processes can be found in Due et al. (1996); Plamondon and Srihari (2000); Mori et al. (1999) and Cheriet et al. (2007). Many standard research databases are available, e.g. CEDAR[2], NIST[3], MNIST[4], or CENPARMI (Suen et al., 1992).

The initial image pre-processing stage of handwriting OCR is very similar to writer identification, especially where historical data is concerned (e.g. binarisation (Gupta et al., 2007) or character segmentation (Bryant et al., 2010)). Some methods however actively remove writer-specific style elements such as slant (Cheriet et al., 2007; Marti and Bunke, 2002) to normalise the writing, making

[1] e.g. online signature data format for interchange ISO/IEC 19794-7
[2] http://www.cedar.buffalo.edu/Databases
[3] http://www.nist.gov/srd/nistsd19.cfm
[4] (digits only) http://yann.lecun.com/exdb/mnist/index.html

it easier to distinguish characters. Ideally, all writer, document, or other style-specific elements would be normalised away, leaving a single canonical form for each character. In this respect handwriting OCR can be framed as an inverse of style-identification problems such as writer identification. These attempt instead to normalise or generalise over the common writing characteristics formed by text content or character distribution, to retain only the idiosyncratic style.

Finally, there are two cases where handwriting OCR and writer identification systems are combined. The first is when using a writer's identity (and associated style profile) to improve character recognition by adapting to their personal style (Brakensiek et al., 2001). The second uses the character recognition rate of a writer-trained OCR system to verify whether a new sample's writer matches. One such writer identification system has been proposed by Schlapbach and Bunke (2004a) and is described in more detail in Section 2.3.8.

2.1.1 Summary

This section has placed automated offline writer identification in the context of its pattern recognition background and the wider field of biometrics, along with summaries of those areas of writing analysis which are closely related, but not directly applicable.

The next sections of this review focus in detail on an analysis of the offline writer identification field. Section 2.2 describes the typical state of the data used in this work, including the datasets in use and the common image pre-processing steps required. Section 2.3 details the range of feature extraction methods present in the writer identification literature.

2.2 Dataset preprocessing

In most image-based text processing, a cleaned, binarised image is expected and most feature extraction methods assume a black ink trace on a white background.

An exception is Wirotius et al. (2003), which hypothesises that grayscale images will contain information on pen pressure, a common online feature. The grayscale images complicate the task of ink trace extraction such that a binarised version is required as a reference, but the performance of the extracted features fails to match that of available static features. This suggests binary images offer a reasonable compromise between loss of writer-specific information and manageable data processing. Further support for this comes from the work in Zuo et al. (2002), where the same experiment was run on grayscale and binary images. The identification rates for binary images were only one percentage point short of the 98% achieved with grayscale, suggesting that there was little additional information to be gained.

Standard algorithms exist for the task of thresholding images, e.g. (Niblack, 1990; Otsu, 1979), and more recent algorithms have been developed for document images (Kavallieratou, 2005) and historical documents in particular (Leedham et al., 2002). However, binarisation is not usually implemented specifically for a writer identification task, as many standard implementations of image thresholding are already widely available[5].

The modern data sets typically used (e.g. IAM (Marti and Bunke, 1999), Firemaker (Bulacu et al., 2003), CEDAR (Srihari et al., 2002), UniPen, or IFN/ENIT (Pechwitz et al., 2002)) tend to have little problem with image noise, but it is a particular problem with historical documents. Data collected recently has the advantage of being designed for the purpose of automatic processing[6], with samples standardised to contain only text of a known layout on a blank background, and to use the same line height, writing instruments, etc. Almost all specifically ask the subject for their natural handwriting. Most historical texts fall at the opposite extreme: the pages are varied in size, content, style, font, writing instrument, and decoration, and are usually noticeably degraded, containing smudges and marks,

[5]For example, ImageMagick (http://www.imagemagick.org)

[6]Details of commonly used datasets are given in Appendix A

variable backgrounds, bleed-through of ink from the reverse of the page, physical warping, and faded or patchy ink traces. Preprocessing on these kinds of dataset tends to require some manual intervention. Example images from these datasets are given in Appendix A.

2.3 Offline Feature Extraction

The bulk of work in writer identification has been in identifying and developing new features. This section will outline the characteristics that describe the range of available features, and provide a qualitative overview of the main groups that have been proposed.

2.3.1 Characteristics

This section explains the terms used to group writer identification features, and also examines the factors in experimental design which affect classification accuracy and the interpretation of identification results. The following sections survey the range of feature extraction techniques currently available, ordered from greatest to least use of writing information: local features are described first (Section 2.3.2), followed by global writing and texture-based features in Sections 2.3.6 - 2.3.7. A comprehensive summary of the corresponding experimental results is available in Appendix B for reference.

Local, Global, and Textural Writer identification features can be grouped by the amount of writing-specific information they require. Textural features use no writing information: they are typically features from image or signal recognition that have been applied to document images without adaptation, such as wavelets (Terzija and Geisselhardt, 2004; Daugman, 2005; Shahabi and Rahmati, 2009; Antonini et al., 1992) and autocorrelation (Bulacu and Schomaker, 2007b; Bulacu et al., 2003; van der Maaten, 2005; Bracewell, 1965). At the other extreme, local

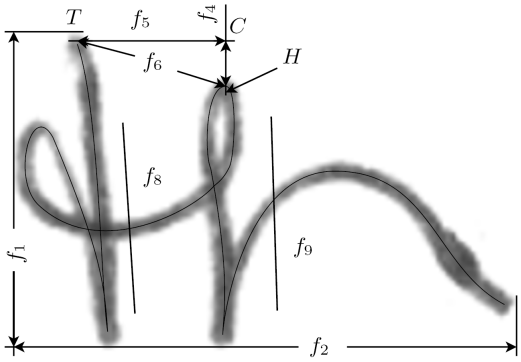


FIGURE 2.1: A manually extracted 'th' from a sample of the CEDAR letter, with structural features marked (Pervouchine and Leedham, 2006)

features are usually measured directly from specific, identified parts of the ink trace, such as strokes or characters. This information tends to come from manual labelling (Srihari et al., 2002; Pechwitz et al., 2002). Between these categories, global features make use of the distinction between ink trace and background, but do not require any transcription of the writing content. They can be extracted or measured from any text. These divisions are approximate, and terminology varies: local and global features are also sometimes described as structural and statistical features respectively.

Text-dependent and Text-independent These terms can be used to describe two related concepts in either feature extraction or experimental methodology. A feature can be text-dependent if it requires a particular text in order to be extracted. This is most often encountered in local, character-based features, e.g. measuring particular parts of a given character, such as the ascender heights or loops (Figure 2.1). Conversely, a text-independent or text-agnostic feature can be applied to any text, regardless of content and often regardless of script, e.g. height of text line.

In methodology, these terms refer to the content of the dataset. Some datasets have a single, fixed-content text which participant writers copy out a given number of times to produce the handwriting samples (Srihari et al., 2002; Nejad and Rah-

mati, 2007). This approach is most common in character-based features (Zhang et al., 2003), but is also occasionally applied to words (Zhang and Srihari, 2003; Zois and Anastassopoulos, 1996). Other datasets have free or varied content samples, either by design (Marti and Bunke, 1999; Pechwitz et al., 2002; Bulacu et al., 2003) or due to the existing data sources (Brink et al., 2007; Bar-Yosef et al., 2007).

The majority of the high performing features in writer identification are text-independent, but their results can still be affected by the use of fixed-content texts. Many people use multiple forms, or allographs, to draw the same letter (Srihari et al., 2003), and selection of a form is usually dependent on the adjacent characters. Having a single possible text for a sample will skew the shape distributions in favour of those graphs and combinations present, especially as most predefined texts are fairly short (e.g. the CEDAR letter (Srihari et al., 2002) is 156 words long, PSI database texts (Bensefia et al., 2005) are 107 or 98 words). This effect, coupled with the exact matching of content across writers, effectively factors out some of the variation that a good feature aims to compensate for. Davis (2007) notes (p. 255) that in forensic use, the main advantage of the ability to request a sample text from a writer is that the content can be identical to the questioned document. Apart from this mention, the issue does not appear to be discussed in the literature, but some indication of its potential effect can be found in Schomaker et al. (2004): using a 15x15-cell grapheme codebook, various pages of the Firemaker dataset are tested. Amongst these, testing the free-content pages of the set yields an identification rate of 70%. The same codebook tested against samples copied from a fixed text by the same writers, again in their natural handwriting, causes a large jump in accuracy to 97%, an increase by a factor of 1.4. In practice in historical use, performance on varied texts is essential as these document collections were not systematically generated.

2.3.2 Local features

One method of automating writer identification is to replicate the processes carried out manually by palaeographers and Forensic or Questioned Document Examiners (FDEs or QDEs). However, these methods have never been aimed at automatic extraction, and their descriptions are often too vague to implement directly. Substantial design choices are usually required to formalise these descriptions into an implementation, so that even those features derived from the same original descriptions may be quite different in practice. For example, Srihari et al. (2002) describe their work as using features similar to those used by document analysts, but include counts of exterior curves and interior contours in the category of measures of writing movement, and counts of sloping components as a measure of stroke formation. Pervouchine and Leedham (2007) also describe needing to apply stroke thinning algorithms to character images due to difficulties in automatically identifying loops. Manual document examination tends to be placed in terms of the interpreted units that are obvious to a person, e.g. characters, rather than the visual artifacts that are actually present, e.g. the ink trace. This means that to directly implement manual features, a very high level of ground truth knowledge of the text in question is usually required, such as information about the bounds of each character and a full transcription of the text. As the problem of general handwritten-image OCR is still very much unsolved, this information is usually obtained manually and/or by designing a fixed source text, as in Srihari et al. (2002); Gazzah and Amara (2007) and Wang et al. (2003).

Local features, whether derived from FDE features or developed independently, are always directly measured. Information extracted in this way describes the structural composition and shape of a character or graph very finely. This can include measurements made of parts of specific characters (e.g. ascenders or loops), or presence or absence of a specific structure (e.g. loop on the descender of a ‘y’). Examples of this type of work include the GSC features described in Section 2.3.3, and the studies of Pervouchine and Leedham (2007, 2006); Sutanto

et al. (2003) and Maclean (2004).

2.3.3 GSC features

A considerable amount of work has been produced using local and structural features which aims to prove that handwriting is a personally distinctive characteristic, capable of distinguishing individuals at the level required to provide evidence in court. Srihari et al. (2002) presents an initial thorough study of the situation, introducing the CEDAR dataset[7] and several feature performance experiments at the document, word and character level. It also applies the gradient, structure and concavity (GSC) features developed for character recognition (Srikantan et al., 1996) to the task of writer identification, a process on which the later studies rely. The features are based on a pixel-level analysis of individual character images. All pixels are initially mapped to a quantisation of their stroke gradient into 12 or 18 directions. The image is divided into a number of regions, often 4×4. Features are drawn from the proportion, variation and presence or absence of certain gradient directions in these regions. Structural features are the presence or absence of some unspecified combinations of pixel gradients. Stroke extraction is performed by thresholding the gradient maps and extracting any remaining connected pixels. The bounding boxes and orientation of these strokes is used to determine concavity. As these features are designed for character recognition, they have been adapted somewhat for writer identification – most notably, the paper proposing these GSC elements describes many real-valued features, whereas Srihari et al. (2002) uses only a binary bit vector. It is not stated whether those used are exactly those described as binary features, or if the real-valued elements have been adapted or encoded in some way.

The initial study has been extended to examine in particular the potential to discriminate between authors on the basis of individual characters (Zhang et al., 2003) and individual digits (Srihari et al., 2003). Single handwritten words (Zhang

[7] Details of this dataset can be found in Appendix A.2 on page 163

and Srihari, 2003) are also studied, and the GSC features are extended here to apply to whole-word images. The data used in these experiments has been extracted manually from around 3000 samples of a copied letter, designed to cover the widest possible range of letter combinations. These works generally do not aim to propose a methodology that could practicably be used in writer identification, and thus the limitations of requiring manual character extraction and having a fixed text are mitigated. The performance of these features is very high, reaching 98% Top-1 correct on a dataset of 875 writers, but the prohibitive manual segmentation and labelling requirements make it unsuitable for most general use.

2.3.4 Slant-based approaches

One aspect of handwriting known to be characteristic is slant, or more generally, the orientations of the strokes a writer tends to produce: angle (or directional) based features are frequently amongst the top performers in writer identification (e.g. 86% Top-1 accuracy from 250 writers (Bulacu and Schomaker, 2006)), and it was found in Schlapbach and Bunke (2005) that removing the characteristic slant from handwriting generally decreased the identification rate.

The distributions of stroke fragment angles have been studied extensively by Bulacu et al. (Bulacu and Schomaker, 2006; Bulacu, 2007; Bulacu et al., 2003), both individually (slant distribution) and in joint distributions of angles that occur together (edge-hinge distributions and direction co-occurrence distributions). This work was extended in van der Maaten (2005) with edge-hinge combinations.

All these features are extracted in similar ways: by performing edge-detection on the ink trace image, then sliding a square window over the sample to locate stroke contours (Figure 2.3). If the central pixel in the window is ink, a search is made to find if there is a continuous edge that meets the window border: if so, the fragment orientation, or angle, is quantised by the window pixel it intersects. The edge hinge distributions follow the same method, except that a measurement is made only if two stroke fragments are found from the central pixel, in which case

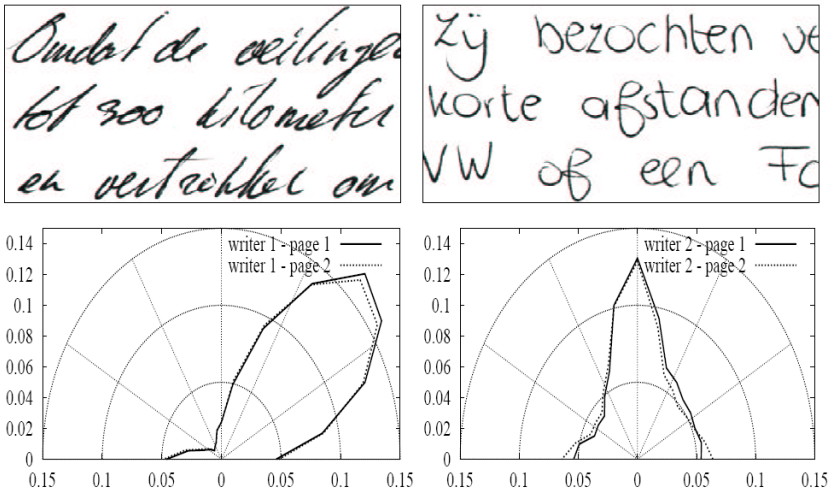


FIGURE 2.2: Slant distributions of two handwriting samples (Schomaker and Bulacu, 2004)

the angle pair is recorded. Direction co-occurrence distributions measure the angle pairs found at either end of a (horizontal or vertical) run length of background pixels, and attempt to extract larger-scale information about changes in writing direction (Schomaker et al., 2004). In all cases, once the counts have been accumulated, they are normalised so that they sum to 1, giving a probability density function (pdf).

The quantisation and dimensionality of the resulting distributions is determined by the size of the window, which also defines the stroke fragment length n. This parameter has been tested for the basic slant distribution for values of 3-5 pixels (Bulacu et al., 2003); the width of the ink trace in the normalised samples used is in the region of 5 pixels. As there is no way of determining the direction in which a stroke was made, only angles between 0-180° are considered, giving these slant features a dimensionality between 8-16. For edge-hinge distributions, values of n between 3 and 9 pixels have been tested (van der Maaten, 2005; Bulacu et al., 2003). As these are joint distributions, the dimensionalities are of the order of the square of the quantisation directions, and range between 104 (at 3 pixels) and 1952 (9 pixels). Direction co-occurrence distributions have been tested with

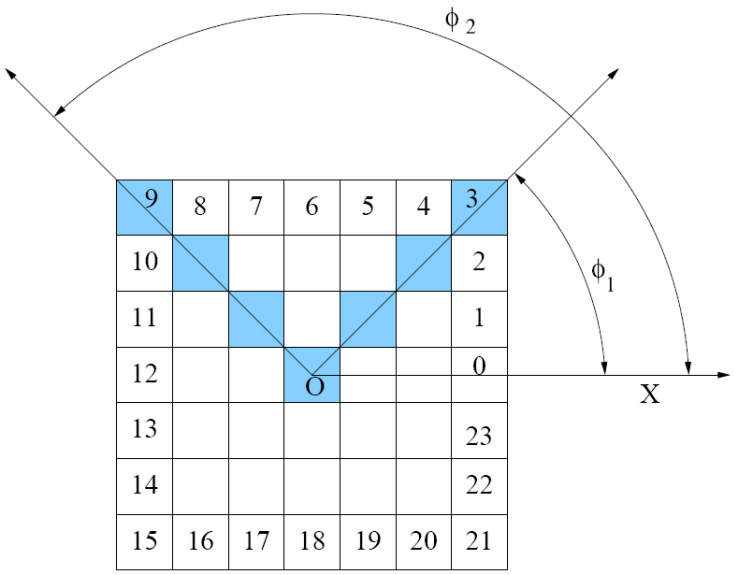


FIGURE 2.3: Measurement of slant and edge-hinge angles (Bulacu and Schomaker, 2003)

4-pixel fragment lengths (Schomaker et al., 2004), with a dimensionality of 144, and accuracies of up to 76% (van Erp et al., 2003).

Edge-hinge distributions

The top performer of slant, edge-hinge and direction co-occurrence distributions is the edge-hinge distribution, which has become one of the top performing global identification features[8] (Bulacu and Schomaker, 2006, 2007b). It was developed from the differentiated slant distribution: the authors state that the increased performance of the differentiated feature vector was due to it capturing information about changes in writing direction (Bulacu et al., 2003). However, this information does not seems to be present in any derivative of a slant distribution: the angle information is aggregated, so detail of the individual transitions that take place in

[8] All three distributions, along with run-length distributions, have been included in WANDA, a writer identification system to aid forensic experts in both manual and automated sample inspection (Franke et al., 2003). An overview of identification systems in use can be found in Srihari and Leedham (2003).

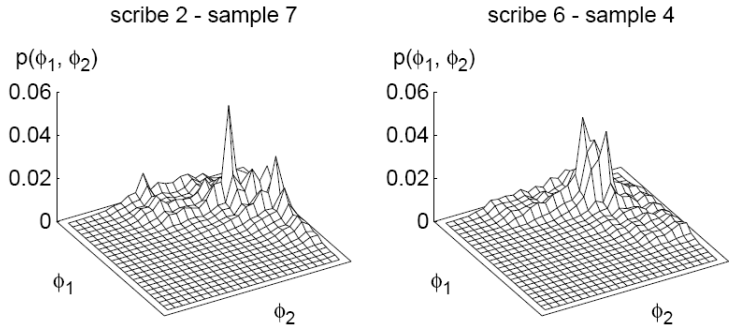


FIGURE 2.4: Edge-hinge distributions of two different writing samples (Bulacu and Schomaker, 2007a)

writing is lost. The two joint distribution features are most likely to contain this information: the same-pixel or same-run length links provide the necessary co-occurrence information, which may explain their superior performance.

Edge-hinge combinations

Edge-hinge distributions were further developed in van der Maaten (2005) to produce edge-hinge combinations. Van der Maaten theorised that hinge pdfs of different fragment lengths contained different information, as they are extracted at different scales. He variously combined hinge distributions of 3, 5, 7 and 9 pixels, and found that concatenating all these into a single feature vector gave the best performance of 81% from 250 writers, despite the resulting feature vector having 3600 dimensions.

2.3.5 Grapheme-based approaches

A different area of work involves shape-based features: those that operate directly on the ink trace fragments produced, rather than extracting a particular aspect of it. All these approaches divide the ink trace into small segments, and cluster them to produce a selection representative in some way of the stroke shapes a writer produces. The segmentation stage typically uses the heuristic of splitting on the

lowest infection points (minima)[9] of the ink trace to produce character-like fragments, as proposed in Casey and Lecolinet (1996). The presence or distributions of these reference shapes in a document form its feature vector.

There are two main groups of work that use graphemes directly – the grapheme codebooks of Schomaker and Bulacu (2004), and writer invariants, proposed in Nosary et al. (1999). In addition, Seropian et al. (2003) and Siddiqi and Vincent (2007) propose similar schemes at a sub-grapheme level, constructing a representative reference base using smaller stroke fragments[10]. These approaches therefore operate on the information that can be gathered by comparing stroke shapes directly. They are divided into those that seek to keep or emphasise the outlier shapes in the set (grapheme codebooks) and those that discard them, retaining the more frequently occurring (writer invariants and stroke-fragment reference bases).

Grapheme codebooks are based on the more general image classification bag-of-words technique (Li and Perona, 2005; Lazic and Aarabi, 2007; Marinai et al., 2010; Woodard et al., 2010). The main differences are a natural rather than artificial segmentation criterion (i.e. a heuristic that approximates characters), and the omission of an explicit feature extraction or selection stage, with the normalised frequency histograms used as features directly.

Grapheme codebooks

The approach of Bulacu et al. requires a reference set of graphemes, against which a similarity profile of a sample's grapheme distribution is calculated. This

[9] See Figure 3.5 and Section 3.3.2 for details of this process and an example of minima-based splitting.

[10] Despite initial similarities, larger 'graphemes', such as the word-level images used in word spotting (Manmatha et al., 1996), are not used. To do so would reduce the effective sample-size from the number of characters down to the number of instances of the selected words. It may also require semantic knowledge of the text content, either as a transcription or through manual segmentation of the image samples. Although a complete word has the potential to be highly writer-specific, there is no guarantee with free-content samples that the words under inspection will appear, and the matching process is highly-dependent on alphabet, language, and font, rendering it unsuitable for comparing multiple scripts across varied domains.

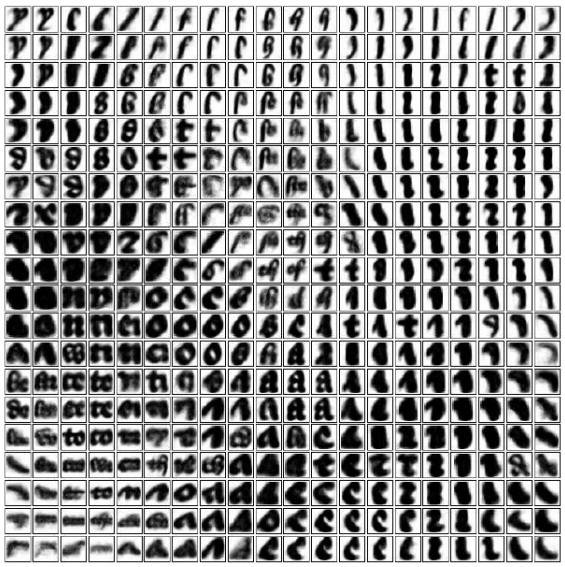


FIGURE 2.5: A grapheme codebook generated using a 2D Kohonen SOFM (Bulacu and Schomaker, 2007a)

reference set, or codebook, is selected by choosing a certain number of graphemes from the entire training set combined. Using the training set allows the codebook to be tailored to describe any style, script, or font that the data might share.

In the initial proposal, a Kohonen self-organising feature map (SOFM) was used to cluster the graphemes, producing a 2D array of selected graphemes which spanned the shape space of the set (Schomaker et al., 2004), but the resulting spatial mapping is not used in the codebook. In Bulacu and Schomaker (2005a), 1D SOFM, 2D SOFM and k-means clustering were shown to give equivalent performance, and van der Maaten and Postma (2005) also showed that random selection produces comparable results. Random selection produces codebooks with slightly different properties to the clustering methods: the probability of a grapheme's selection reflects the underlying training set distribution, whereas clustering-based selections remove the repetition of the most common elements and weight towards outliers – frequency of occurrence is not a factor. However, the magnitude of this effect is unclear, as codebook sizes (typically 50-200 graphemes) may not be large

enough to reflect the differences significantly. The work in Ghiasi and Safabakhsh (2010) introduces a new segmentation criterion which outperforms the usual ink-trace minima segmentation used on smaller Arabic text sample sizes.

Given a complete reference codebook, a sample's grapheme distribution is calculated by binning each grapheme in turn into the closest match present in the codebook. The remaining frequency distribution has dimensionality equal to the codebook size and is normalised before being used as the sample's feature vector. Larger codebooks have the potential to describe a sample more accurately, but suffer the curse of dimensionality. Codebook sizes of around 400 are often found; a size of 1089 was originally tested (Schomaker and Bulacu, 2004). In Schomaker et al. (2004), performance was found to plateau or tail off slightly beyond a codebook of size 225 for Latin texts. No studies have been performed to analyse the relative performance of graphemes or grapheme types (e.g. rounded, single-stroke, complex, large). Considering that the best-performing selection method currently available performs equivalently to random selection, this seems an area which could benefit from further analysis.

Performance of grapheme codebooks is good, with accuracies up to 100% on Arabic texts (Ghiasi and Safabakhsh, 2010), but more usually around 80% on a 650-writer, text-independent dataset (Bulacu and Schomaker, 2007b). In Schomaker et al. (2007) it was further found that smoothing out the binning counts for graphemes gave a significant performance boost, increasing Top-1 classification accuracy from 71% to 82%. In smoothing, a smaller portion of the 'count' allocated to a bin (a matched codebook grapheme) is split between its n closest neighbours, as determined by shape similarity (in strength of match, not SOFM layout-based proximity).

Writer invariants

The original proposal for writer invariants did not require a global codebook,

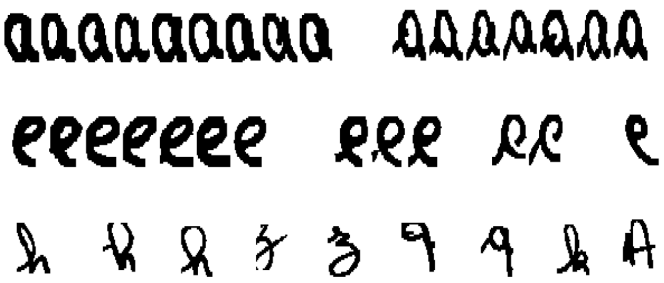


FIGURE 2.6: Individual graphemes from invariant and non-invariant (discarded) clusters (Bensefia et al., 2002)

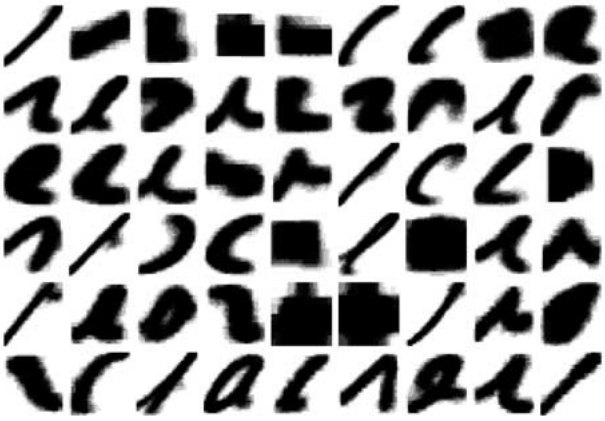


FIGURE 2.7: Graphemes composed into invariant clusters (Bensefia et al., 2005)

and was designed with image retrieval and data compression in mind: the training set images are indexed by a feature vector to be retrieved on same-writer queries. To produce the feature vector, a clustering algorithm is run over the graphemes generated from the sample. The sequential algorithm used (Bensefia et al., 2002) includes a stochastic element, so this process is repeated. Graphemes which are clustered together on every run are grouped into an invariant; all others are discarded. Image correlation is used as the similarity measure on the grapheme bitmaps at this and all later image comparison stages.

In the information retrieval context, Bensefia et al. (2002) defined a similarity measure between two sample documents D and T in terms of a normalised sum of the similarities of each of D's graphemes to its best-matched grapheme

from T. Query document D is then assigned the writer of its closest matching training set sample. As this calculation requires computing the similarity between every grapheme in the query sample and every grapheme in the entire training set, the authors compress the training set documents by replacing their grapheme sets with their invariant sets. This caused no loss of precision in retrieval (98% from 88 writers). This result may suffer from a favourable fixed-text bias in the dataset, but it indicates that writer-specific information may be concentrated in the typical rather than the atypical shapes produced. This result is an interesting counterpoint to the use of grapheme codebook clustering selection, and also the χ^2 distance metric used by Bulacu et al. for computing similarity between distributions. The authors specifically state that this metric weights the lower-probability areas of the distribution (Bulacu and Schomaker, 2007b), i.e. the less common elements. They found this metric outperforms several others including Euclidean, Hausdorff, and Bhattacharya, although Hamming distance apparently also performs well (Schomaker et al., 2004; Schomaker and Bulacu, 2004).

Later writer invariants-based proposals have drifted much closer to the grapheme codebook approach, in particular using a reference set of invariants computed from the entire training set, and using a similarity measure based on binning/thresholding rather than raw correlation (Bensefia et al., 2005). The switch from individual writer profiles to a combined reference set was presumably made to reduce the number of comparisons required in a query/identification calculation. However, the authors do not state how much of a reduction pooled invariants produce and as this change was made in conjunction with others, it is impossible to isolate any effect this may have had on identification performance.

Stroke fragments

Seropian et al. (2003) and Siddiqi and Vincent (2007) both propose codebooks composed of lower-level stroke fragments that are much smaller than the graphemes used above, but make use of them in very different ways. Siddiqi and Vincent

(2007) tiles the ink trace with small windows to achieve this segmentation, and tests various window sizes. Unfortunately no reference to the dimensions of the image or writing is given, which makes these hard to interpret. The codebook, or reference base, is composed of a representative image from each fragment cluster above a certain threshold size. This is closest to the writer invariants approach which also discards outlying graphemes. However, no distributions are calculated - similarities between reference bases are calculated directly by maximising the total image correlation between their train and test fragment sets. The approach gives very good results of up to 94% on a fairly small (50 writers) IAM dataset.

Seropian et al. (2003) appears to use a representative set of stroke fragments to reconstruct the handwriting image, in the manner of an Iterated Function System (Hutchinson, 1981; Barnsley, 2000). Similarity between these reference bases is determined by substituting fragment images from one base into another, and quantifying the quality of the resulting reconstruction of the original image using the peak signal-to-noise ratio. Identification accuracy appears to be good but is reported imprecisely ($> 85\%$) for a 20-writer dataset.

2.3.6 Run-length distributions

The first feature proposed for automatic writer identification in practice was the run-length distribution (Arazi, 1977). This is a global, purely statistical analysis of an image that involves counting the lengths of continuous runs of pixels of the same colour (usually black and white), and converting the frequencies of occurrence into a probability density function. Run lengths can be computed either horizontally or vertically, and on background (white) or ink trace (black) pixels. Arazi states that background runs will be more informative, as they can convey information about the distributions of inter- and intra-character spacing, whereas ink run distributions convey mainly information about stroke widths. This is plausible, as in cases where the influence of the writing instrument cannot be factored out, any writer information will be very noisy. The only test of this assertion ap-

pears in Bulacu et al. (2003), which shows both vertical and horizontal run-lengths perform better on background than ink pixels. The run-length still frequently appears in recent work, and although its individual performance as a feature is not that high (20-30% in the best cases (Brink et al., 2008; van der Maaten, 2005; Bulacu et al., 2003)), it has proved useful in combination with others (Bulacu and Schomaker, 2006; van Erp et al., 2003).

2.3.7 Textural features

This class of features borrows techniques directly from general image analysis, examining each handwriting sample without making use of the information that writing is present. They usually draw out global aspects of the writing style, but can be susceptible to distraction by page- and text-based differences, e.g. irregular layout or line spacing, which often do not reflect personal writing style. To mitigate this, the images are often pre-processed to make them more uniform in spacing (Ubul et al., 2009; Shahabi and Rahmati, 2009; Zhu et al., 2000; Fornéz et al., 2009).

Wavelets and filters

Although wavelets are used in many image processing applications (e.g. Terzija and Geisselhardt (2004); Daugman (2005)), they do not in general seem to be a useful feature in writer identification. Many classes of wavelet have been tested, including Haar, Odegard, Villasenor and Daubechies (Schomaker and Bulacu, 2004; van der Maaten, 2005), but they have never reached a useful level of performance. However, the Gabor wavelet appears to be an exception: Schomaker and Bulacu (2004) suggests its periodicity may give it an advantage in picking out angular information, and it appears frequently as a high-performing feature in non-Latin scripts, particularly Arabic and Chinese (Shahabi and Rahmati, 2006, 2009; He et al., 2005; Zhu et al., 2000). Shahabi and Rahmati (2006) tests Gabor energy features, filters and various transforms on a dataset of Farsi handwriting,

with performance considerably better than those for most wavelets. Said et al. (1998) also use Gabor filters with similar results, however both papers use datasets that are small (25 and 20 writers respectively) and no information is given on how the results might scale with increasing numbers of writers.

Grayscale co-occurrence matrices

First proposed in Haralick et al. (1973) for general image classification, grayscale co-occurrence matrices (GSCMs) are usually employed in binary form only for writer identification. They are calculated by considering the frequency of occurrence of each grayscale value in the Moore neighbourhoods of all pixels with a particular value, forming a square matrix whose dimensions reflect the quantisation of grayscale values in the image. The neighbourhoods most used are of radius or width 1, i.e. the immediately adjacent pixels only, but larger widths can also be used. For binary images, this will be 2×2.

GSCMs of several widths have been used as a reference features in Said et al. (1998) and Shahabi and Rahmati (2006) against various Gabor-based filters. They gave reasonable performance on small datasets, though are usually outperformed by the filter-based features.

Autocorrelation

Horizontal autocorrelation is used as a measure of how well writing resembles itself, extracting information on regularity in handwriting and evenness of spacing between vertical strokes. This is measured by taking each row of pixels in turn and shifting it against itself repeatedly, calculating each time the correlation or Hamming distance between the matched pairs. With a top identification rate of 25% on 150 writers (Schomaker et al., 2007) and 13% on 650 writers (Bulacu and Schomaker, 2007b) its performance is poor, although like most texture features it has not been tested in combination. Vertical autocorrelation is not used, as its values will be determined mainly by the content of the writing, i.e. the characters

that line up in each column.

2.3.8 Hidden Markov Models

Schlapbach and Bunke (2004a) propose a writer identification system based on the output of a pre-existing handwritten OCR system (Marti and Bunke, 2002). By composing Hidden Markov Models (HMMs) for character and word recognition, a model is trained for each writer's style from sample data. The features used are extracted from a sliding window passed horizontally over binarised text lines, but as these are designed for character recognition rather than writer identification, they do not fall into the usual categorisation. The text is processed before the feature extraction stage by normalising the average character width, scaling the text vertically and removing the ink trace slant. Although these may lose some writer identification information, they were initially retained as they improve the readability of the text, which this system depends on to give accurate results. A later comprehensive study into these normalisations concluded that slant correction in particular decreased the writer identification rate (Schlapbach and Bunke, 2005); subsequent applications of this system apply only vertical scaling (Schlapbach and Bunke, 2007b).

For identification, all models are run over an unknown sample, each outputting a transcription of the text and a likelihood score of that transcription. As correctly recognised words should have a significantly higher score than incorrect ones, the likelihood should be a good measure of the accuracy of a model over a sample. A model trained to recognise a writer's style should output more accurate transcriptions, thus the best writer matches are determined by the highest likelihood scores. Performance of this system is very good, at 97% for 100 writers (Schlapbach and Bunke, 2007b) but the prohibitive training requirements (including a full transcription of all samples) make it unsuitable for most practical applications.

Some of these issues were addressed by a significant pruning of the original system down to the Gaussian Mixture Models (GMMs) used in HMM training

(Schlapbach and Bunke, 2006a,b). This drastically cuts down on the training requirements, as no character or word models are needed. This in turn removes the requirement for a ground truth transcription of each sample. The identification performance is also slightly increased to 98%.

Both the HMM and GMM approaches require training a model for each expected writer. This is a disadvantage in any system that continually adds to its classified document set, as an expensive retraining step must occur to make use of each update. (This is particularly important in scribal identification, as new samples may be in very different styles or fonts to the known data, which will significantly extend the class boundary.) The problems of handling unknown writers in this system can be alleviated with an accept/reject threshold from writer verification (Schlapbach and Bunke, 2006a, 2004b).

2.3.9 Non-Latin scripts

This section will briefly review the main work in writer identification for non-Latin scripts.

Although the majority of work involves Western and especially Latin alphabet datasets, there have also been several, albeit more isolated, studies into writer identification for other scripts. The performance of features across different scripts is of interest when applying features to a new domain, in this case medieval English and Latin scripts. Although the majority of current work is text-independent, the distribution of typical letter shapes in a script can affect the performance of a feature. There has not been much work in testing features across scripts: those proposed tend to be designed for a particular domain, and are generally very different across alphabets. The most significant exception to this is (Bulacu et al., 2007a), where grapheme codebooks, edge-hinge distributions and run lengths are applied directly to an Arabic-script dataset. There was a small performance drop compared to the typical accuracy on Latin datasets, but overall performance was comparable, suggesting that these features extract information common to a vari-

ety of handwritings.

Arabic and Chinese scripts feature most prominently in the literature. The main reference database available for non-Latin scripts is the IFN/ENIT[11] database of handwritten Tunisian placenames (Pechwitz et al., 2002). It is used for both writer identification (Abdi et al., 2009; Bulacu et al., 2007a) and handwriting recognition experiments, most notably the ICDAR Arabic Handwriting Recognition Competitions (Märgner et al., 2005; Märgner and El Abed, 2007, 2009). A smaller Farsi dataset has been used for work involving Gabor filters (Shahabi and Rahmati, 2009; Nejad and Rahmati, 2007) and as a supplement in further developing the grapheme codebook feature (Ghiasi and Safabakhsh, 2010), and custom datasets have been used to test gradient features (Sadeghi ram and Moghaddam, 2009), wavelet transforms (Gazzah and Amara, 2007), SIFT features (Woodard et al., 2010), and Gabor-based graph matching (Helli and Moghaddam, 2009). Shahabi and Rahmati (2006) apply Gabor-based features using a fixed-content dataset from 25 people, although the written pages were divided into non-overlapping blocks. The Gabor filters, transforms and energy features were compared favourably against the grayscale co-occurrence matrix proposed for general texture recognition in Haralick et al. (1973).

Wang et al. (2003) draw features from individual Chinese characters. These directional element features are usually used for character recognition, but are applied here in conjunction with dimensionality reduction to determine authorship. Two datasets were tested, the main consisting of a large training set drawn from over 600 writers (although most contribute only a single character) and a test set composed of 6 samples of 20 characters from 27 writers. All characters are tested individually, i.e. only like characters are ever compared, making this a text-dependent approach also. Gabor filters have also been tested (Ubul et al., 2009; Zhu et al., 2000), including a wavelet variation using a Generalised Gaussian Density (GGD) model for Chinese characters in He et al. (2005). Following

[11] Details of this dataset can be found in Appendix A.7

the texture approach of Said et al. (1998), they take 64-character samples from 10 people, using one sample for training and another for testing. The two methods show similar performance, but the GGD approach shows a substantially reduced computation time.

Beyond these, Bar-Yosef et al. (2007) examine binarised images from a small dataset (34 images) of historical Hebrew manuscripts. Much of the paper focuses on the image processing and character segmentation required, which is typical of historical datasets. Character segmentation is somewhat easier in Hebrew calligraphy as characters do not join together. Their approach is highly text- and alphabet-specific, as it extract features from the letters Aleph, Ain and Lamed only: using the convex hull of each character image, features such as central moments, aspect ratios and estimations of curvature are combined into a single vector.

Zois and Anastassopoulos (1996) use first-order central moments and morphological openings as features. The dataset is composed of 20 writers, who contribute eight samples of each of four Greek words. These are processed individually to determine the moments, and the vertical projection profile of each word is incrementally truncated to obtain the morphological opening features.

In addition to these studies, many use databases of Western but non-English scripts. The PSI database[12] is composed of samples written in French, and the NFI[13] and Firemaker[14] datasets are in Dutch. Although these languages are strongly Latin-based, they contain digraphs not found in English. These are joined letter pairs that form a single written character, effectively extending the base alphabet, and both languages use diacritics above or below letters. Some work is also beginning in writer identification from handwritten musical manuscript pages (Fornéz et al., 2009; Marinai et al., 2010).

[12]Details of this dataset can be found in Appendix A.3
[13]Details of this dataset can be found in Appendix A.5
[14]Details of this dataset can be found in Appendix A.4

2.3.10 Historical work

There is comparatively little writer identification work on historical data. Although interest in this area is increasing, much of the current work focuses on handwriting recognition (Fischer et al., 2009, 2010) or document image processing (Likforman-Sulem et al., 2007; Leedham et al., 2002; Bulacu et al., 2007b).

As mentioned in Section 2.3.9, there has been some work into historical Hebrew manuscripts with good results, but as it relies on specific Hebrew characters the approach is not applicable outside that alphabet. For Latin-based scripts, Bensefia et al. (2003) tested the writer invariants proposal on the PSI database and a dataset drawn from the 19th century correspondence of Emile Zola. They note a number of image processing difficulties with the historical data, and conclude that these probably contributed to its significantly lower performance. A medieval Italian dataset was tested with features composed of statistics calculated from a Zipf power-law curve, adapted to characterise a two-dimensional image texture (Pareti and Vincent, 2006). A peak result of 80% was achieved, but the dataset itself is not fully described and its size is unknown.

Work in the area of medieval manuscripts has tended to focus on databases and modelling support for the manual identification process. Examples of this include the Medieval Scribes website[15] which offers a searchable index of character exemplars for about 80 scribes, and the System for Palaeographic Inspections (Ciula, 2005) for the analysis and comparison of manuscript letterforms.

The existing scribal identification work in medieval English manuscripts uses very small datasets. Stokes (2007) initially applies run-lengths, autocorrelation, edge- and hinge-directions to six images by two scribes with promising results. The remainder of the paper describes development of a tool to support data-entry of manually-described features for clustering.

Bulacu and Schomaker (2007a) applies edge-hinge distributions, grapheme codebooks and run-length distributions to images from 10 scribes, finding perfor-

[15] `www.medievalscribes.com`

mance equivalent to a 900-writer modern dataset (Bulacu and Schomaker, 2007b); however the historical dataset uses 2-15 samples (usually 6 or 7) per writer, whereas the modern dataset has only two. An interesting exception to this equivalence is the run-length distribution feature, which performed far better on the historical dataset than the modern one. This may be an artifact of the increased number of samples per writer or the scaling effect of a small dataset.

2.3.11 Feature Selection and Extraction

Common subsequent stages of feature refinement within the wider pattern recognition field are feature selection and feature extraction. Feature selection techniques choose subsets of an original set of features with two aims: improving the classification rate by discarding irrelevant or poor features, or reducing the number of features as far as possible without decreasing the existing classification rate. Feature extraction methods combine existing features in some way to create new features which better describe the input data. Again, this can be with the objective of reducing the number of features and/or increasing accuracy.

As the majority of the work in writer identification has focused on developing or deriving new features, these techniques are currently used more rarely within the writer identification literature[16], with a few notable exceptions.

Of these two stages, some form of feature selection is more common. Sequential selection methods start with either the complete set of features (Sequential Backward selection, or SBS) or the empty set (Sequential Forward Selection, or SFS), and remove the worst-performing feature or add the best-performing feature on each iteration, before reclassifying the data with the updated feature subset. The 'floating' variations (Pudil et al., 1994) combine these approaches, e.g. Sequential Floating Forward Selection (SFFS) adds the best-performing feature at each iteration, but also checks to see whether performance of the updated set

[16]e.g. neither the surveys of (Plamondon and Lorette, 1989) nor (Sreeraj and Idicula, 2011) consider this

could be improved by removing a previously-chosen feature. Fornéz et al. (2009) test various sequential selections and finds the best results with SFBS, reducing the original set of 92 features to 11, with a small drop in accuracy.

Genetic Algorithms (GAs) are a class of stochastic optimisation techniques which takes an initial population of solutions (in this case, various subsets of features) and tests their performance, keeping the best-performing each iteration and mutating and recombining them to produce new potential solutions. Pervouchine and Leedham (2007) uses GAs to group 31 features measured from the grapheme 'th' into those considered 'Indispensable', 'Partially relevant', or 'Irrelevant'.

Of the feature extraction methods[17], Principal Component Analysis (PCA) is the most commonly applied technique. It seeks to describe most of the variation in the training data in as few dimensions as possible (without taking class label information into account). The same number of resulting dimensions are output, but are ranked according to the variation accounted for. Typically the top 10% of PCA dimensions will represent the data well. Bulacu et al. (2003) briefly mentions applying PCA to confirm the "excessive dimensionality" of the feature vectors they have developed, and van Erp et al. (2003) states that "...PCA analysis confirmed that a reduction to 10% of the original dimensions may still yield reasonable results.".

Linear Discriminant Analysis (LDA) is another common method used to find the most writer-discriminating directions in a feature space. It can be used a classifier in its own right, finding the best linear splitting planes between the training data. LDA operates on the assumptions that the class covariances are the same, and that the feature data are normally distributed. It seeks to best separate the training data by trading-off two measures: minimising the within-class variance (i.e. mapping each class's data in as tight a cluster as possible) and maximising the between class variance (i.e. placing different classes as far from each other as

[17]Details of the feature extraction methods summarised here can be found in e.g. Hastie et al. (2009)

possible). As with PCA, the direction in which the ratio between these measures is maximised has the highest weighting, leading to ordered output dimensions. As LDA models the splitting planes between classes, the number of output dimensions is one less than the number of input classes. Wang et al. (2003) applies both PCA and LDA to the Directional element features common to Chinese writer identification, with excellent individual-character results near 100%.

Schlapbach et al. (2005) investigates many of these techniques, including PCA and Multiple Discriminant Analysis (a variant on LDA designed for the multi-class problem) for feature extraction, and a GA and various sequential algorithms for feature selection. Several techniques manage to reduce the number of features and increase overall identification accuracy; of these, MDA performs best in both categories.

Independent Component Analysis is, like PCA, an example of blind-source separation in that it does not require the classes of the data. Unlike LDA, it makes the assumption that the input feature data are not normally-distributed, using this to search for the feature combinations and weightings most likely to have produced the observed feature data. Ubul et al. (2009) applies a combination of PCA, ICA, and a genetic algorithm to Gabor-filter features, improving identification accuracy from 83.4% to 92.5% whilst reducing the number of features used from 96 to 20.

2.3.12 Feature Identification Performance

Due to the wide variation in experimental setup, it is very difficult to compare feature performances precisely. Nevertheless, some general conclusions can be drawn. The top-performing structural features (GSC) outperform the top global or statistical features (GMMs, edge-hinge combinations, grapheme codebooks), and maintain this performance over a large dataset (875 writers). Due to the time-consuming manual character segmentation that must be performed, all studies use the same (CEDAR) dataset. This consists of fixed-content samples which may

positively skew the results by a significant amount[18]. Of the text-independent features, the GMM approach of training each writer gives the best results, at a maximum of 98% on 100 writers, but this does involve experimental parameter adjustments over the training set. The largest datasets used with text-independent features are in Bulacu and Schomaker (2007b) and Bulacu and Schomaker (2006), of 650 (IAM dataset) or 900 writers (IAM and Firemaker). Several features have been tested on these, the best-performing being the edge-hinge distribution at approximately 80%. The related direction co-occurrence distributions are also reasonably good, but are generally outclassed by the other angular features. The edge-hinge combinations of van der Maaten (2005) boost the performance of the single hinge distribution in a comparable implementation, but do not exceed the highest performance of 84% recorded for this feature. This may be due in part to the ceiling effect of smaller datasets, where a few misclassifications have a noticeable effect on accuracy figures. Grapheme codebooks are of comparable performance to edge-hinge pdfs over many datasets, and the high performance of writer invariants and stroke-fragment reference bases suggest this is a robust approach.

At a high of 69%, the brush feature is surprisingly accurate, but is only practical for datasets where the writing instrument is standardised.

Most texture-based features do not perform very well for the writer identification task, and are generally tested only on small datasets. In particular, wavelets (with the exception of Gabor wavelets) have shown near-uniformly poor performance for the writer identification task, although the related filter-type features show more promise. Autocorrelation performs little better, but its accuracy seems to hold well over increasing numbers of writers. However, due to potential implementation differences, comparison between different experiments is unreliable. Entropy, Hough features and fractal dimension measurements also perform very poorly, to the point where they are unlikely to be of any use in writer identification,

[18]For details of this effect see Section 2.3.1

even in combination.

For further information, Appendix B contains a comprehensive listing of the experimental results available in the offline writer identification literature.

2.4 Summary

This review has considered offline writer identification as a pattern recognition problem, placed in the field of biometrics and closely related to other forms of writing analysis such as signature verification and optical character recognition. The datasets and image processing techniques in common use were briefly surveyed, before considering in detail the area of writer identification which occupies most of the literature: the design and extraction of writer-indicative features.

The full range of techniques in use was described, from those which require complete text transcripts to those which operate on images regardless of text content. It is clear that there are few consistent methodological approaches, which makes it difficult to accurately compare feature performance between experiments. However in general, the features which employ most writing-specific information give the highest identification performance, but offer the least flexibility in adaptation to new corpora.

The majority of the work has been conducted on Latin scripts, but Arabic and Chinese scripts are a developing area. Almost all published work focuses on contemporary data, with only a few scattered applications to historical images.

The next chapter of this work describes and explains the datasets, processing methods, feature extraction techniques, and experimental methodology chosen to support the work in this thesis.

Chapter 3

Data Processing and Experiment Methodology

3.1 Introduction

This chapter describes and explains the significant choices of data and writer identification method used in the technical work of this thesis. The focus of this work is the analysis of historical data, in particular a medieval English manuscript dataset. A typical modern dataset drawn from the standard IAM database is examined in parallel to observe where the two sets respond differently. Section 3.2 describes the data characteristics and details the preliminary work done in processing the datasets.

The grapheme codebook was chosen as the writer identification method for several reasons:

- Excellent performance as an individual feature
- Adaptability and automatic specialisation to different scripts and text styles
- Does not require transcripts, OCR, or manual text labelling
- Well-established method, tested extensively and independently

- Visually comprehensible, and explainable by analogy to manual writer identification procedures

As described in Section 2.3.5, it is one of a number of similar approaches to writer identification which aim to characterise the writing style of a sample by considering the distribution of different ink trace shapes present in the text. Section 3.4 details the stages in the codebook process, discussing the variations in implementation found in the literature.

The final part of the chapter draws on the codebook discussion to define a consistent experimental methodology. This includes the measures taken to ensure that the results obtained are reliable, and (as far as is possible) comparable to earlier studies, and includes some early preliminary experiment results.

An overview diagram of the complete process is given in Figure 3.1.

3.2 Data Preparation

Most contemporary datasets for testing writer identification algorithms are created under standardised test conditions that do not reflect non-laboratory datasets: there are typically a large number of writers generating text that meets fixed criteria, using standardised writing instruments. In non-laboratory data, the text content, quantity, and production usually varies widely between writers, and the document reproductions may be very noisy. These qualities can be particularly pronounced in historical datasets, and furthermore there is no possibility of generating additional data.

However as the volume of data being digitised grows, automated analysis of historical documents is increasingly important. Historical datasets often differ in many aspects from modern benchmark datasets on which features are designed and tested, and techniques which are useful when handling modern datasets may therefore be unsuitable for application to historical data.

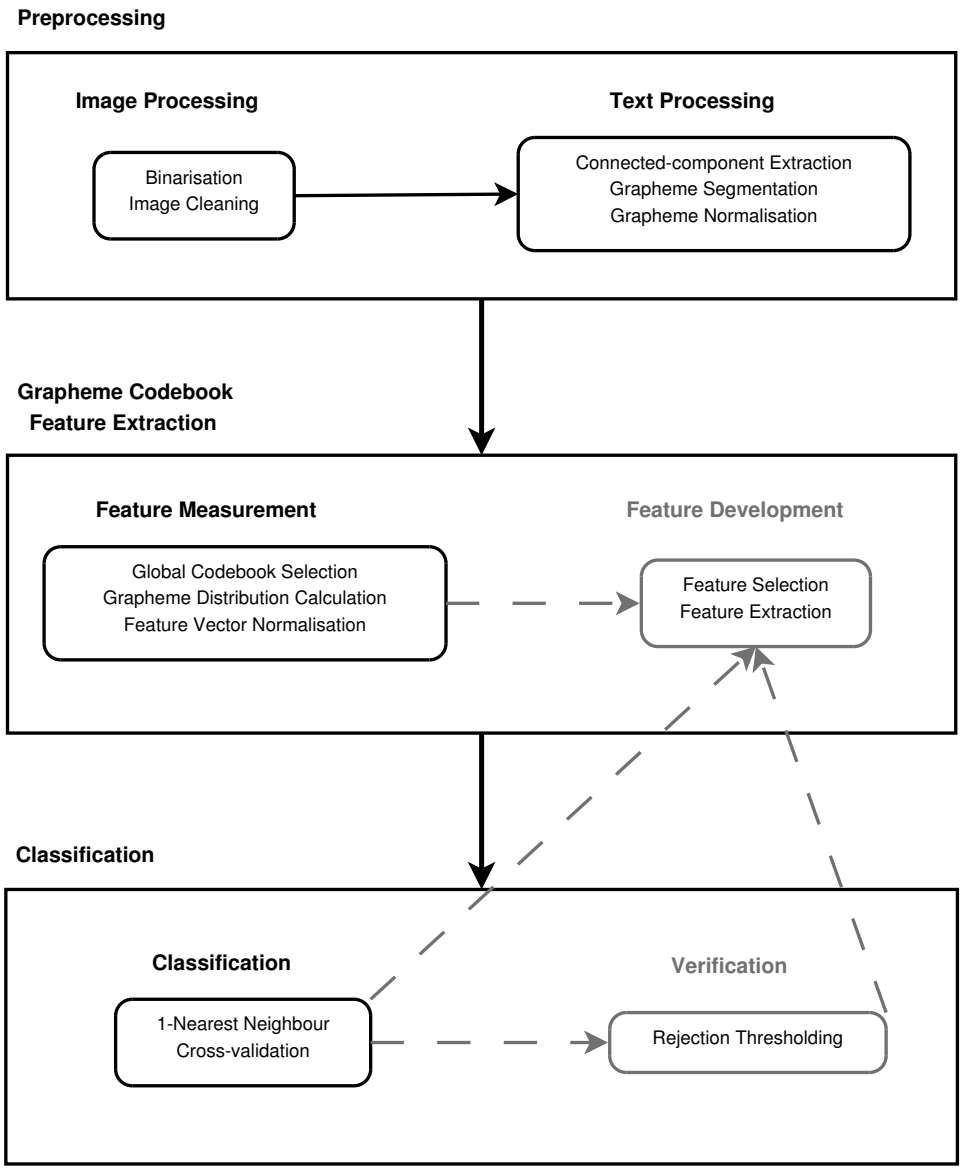


FIGURE 3.1: Overview of the grapheme codebook identification process. Sections in black are required for the grapheme codebook process. Sections in grey are optional stages typically carried out repeatedly during feature development, incorporating information fed back from earlier classifications.

In order to study this, two datasets will be examined: a widely-used contemporary dataset, and a new dataset consisting of photographs of medieval English manuscript pages.

This section describes the standard IAM dataset and the medieval manuscript dataset collected and examines the dataset-specific processing that was applied to each.

3.2.1 IAM

The IAM database is a collection of grayscale PNG images of varied-content text. The images are available in full pages, or at the text line or word level, and are fully labelled. The content of the samples is freeform English text. Each test page contains about a paragraph of text, using a separate guideline sheet to ensure that the lines are horizontal and well-spaced. More details can be found in Marti and Bunke (1999) and Appendix A.1. The modern handwriting dataset is drawn from this database and consists of the 93 writers made available from the 100-writer identification set (Schlapbach and Bunke, 2007b). The images are greyscale, containing a single line of text segmented from a copied varied-text paragraph. Image noise is virtually non-existent – standardised recording forms were used, scans are uniform and high-quality, and text lines are cleanly separated, making it an excellent baseline for comparison.

3.2.2 Medieval

The historical dataset contains approximately 400 full- and part-page images from Middle-English manuscripts, written by 43 scribes. There are between one and 52 images attributed to each scribe; identification of each image was provided by University of York Professor of Medieval English Palaeography, Linne Mooney. As described in the Introduction, several scribes may contribute to a single manuscript. However, the page images here have all been positively attributed to a single writer

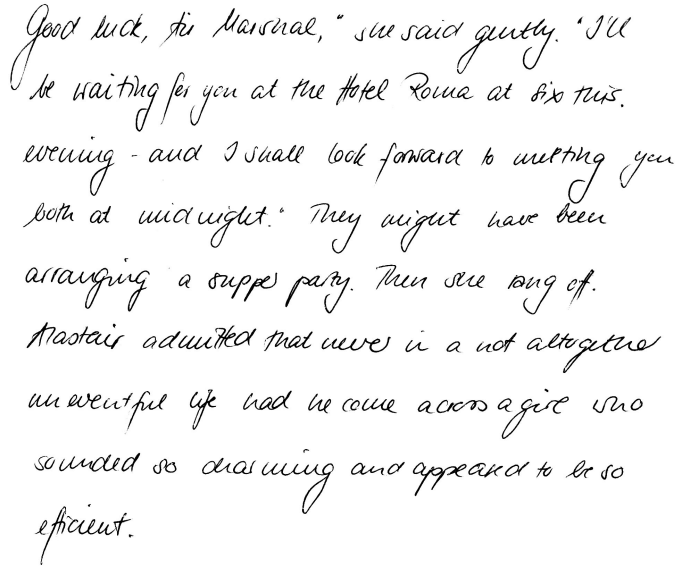


FIGURE 3.2: Full page IAM dataset sample (Marti and Bunke, 2002)

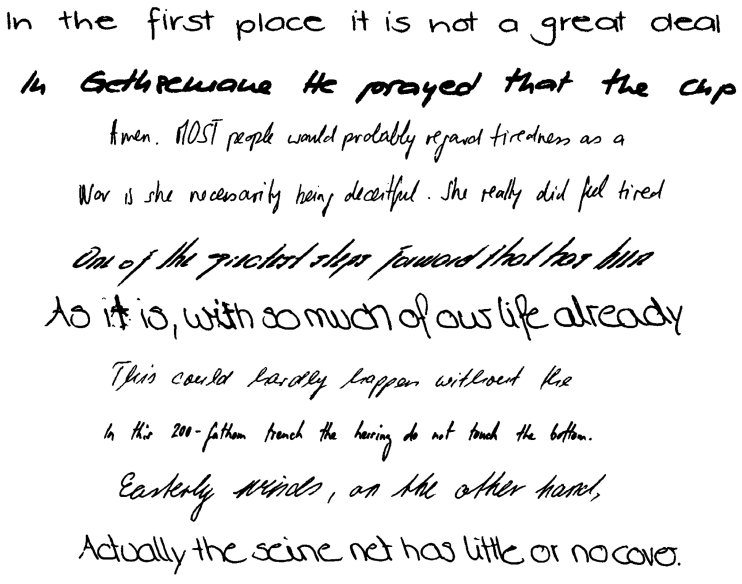


FIGURE 3.3: Selected lines from IAM database handwriting samples (Marti and Bunke, 2002)

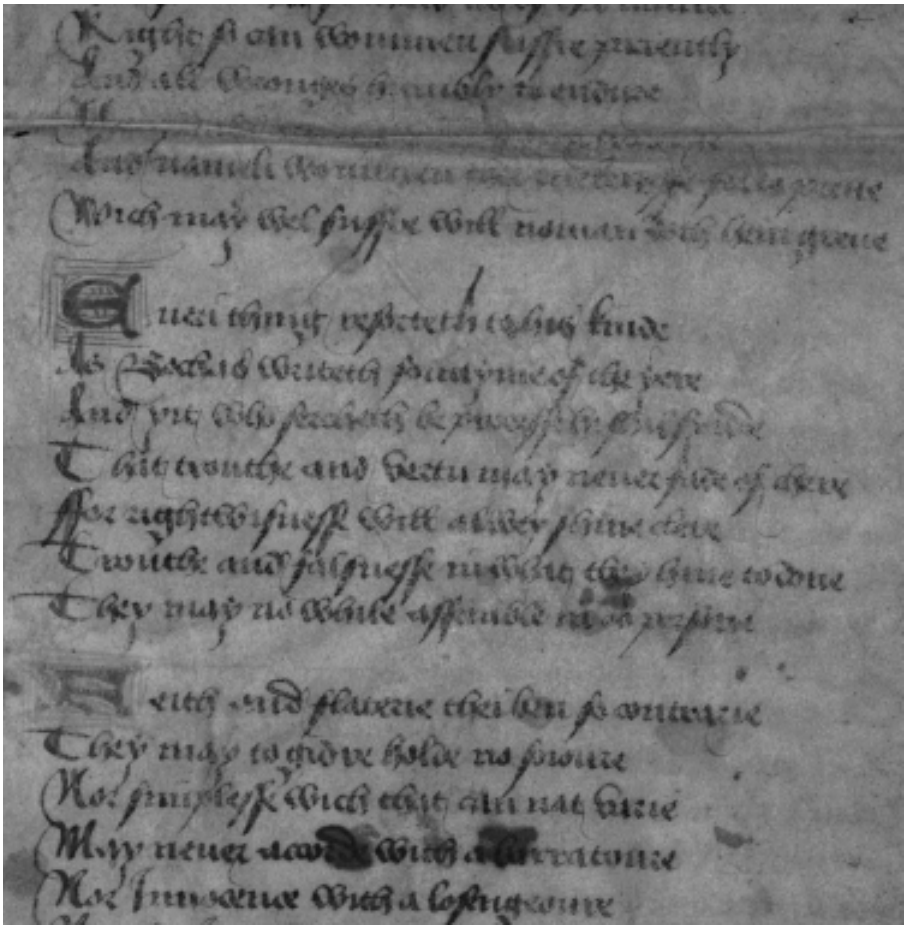


FIGURE 3.4: Sample image from the medieval scribes dataset

to enable accurate system training.

The dataset is very irregular, and image noise levels are high. The ink trace is often broken and faded, text lines can be curved or overlapping, and usually both ink and background vary in colour due to aging or staining (Figure 3.4). Even where the document is well-preserved, the script within a page can change size, layout, and font. The images also vary in size and resolution, from archival quality to samples taken with a handheld digital camera.

3.3 Datasets

The grapheme codebook method requires binary graphemes as input. This section describes the processing applied to convert the original images into the required format. A brief outline of the necessary stages is given (covering the 'Preprocessing' section of Figure 3.1), followed by the detail of the process applied to each dataset and the implementation which supported it, comparing the preprocessing required for the modern and medieval data.

3.3.1 Image Processing Stages

As covered in Section 2.2, there are several image processing stages that generally need to be applied in order to make use of the image data in writer identification. For these experiments, the input data format is in graphemes, so the following stages are necessary and are briefly summarised below: cleaning, binarisation, connected-component extraction, and grapheme extraction. The graphemes have been represented as bitmaps, as this format offers a good trade-off between complexity of processing and use, and the potential classification accuracy (Bulacu and Schomaker, 2005a). No further processing steps (such as contour extraction) have therefore been applied.

Cleaning This stage is applicable where no suitable threshold can be found to binarise the original images directly, either because they make use of colours that map too closely together in greyscale, or because the image is very noisy, or contains non-text elements.

Binarisation The vast majority of offline writer identification techniques work with binary input images, rather than colour or grayscale. Although it appears that no comprehensive studies have been done, available results suggest that despite the information loss, this is the most useful format to work with. Although grayscale may be appropriate for some features, e.g. attempting to reconstruct pen pressure (Wirotius et al., 2003), for the most part it makes the resulting images too complicated to process further[1].

Connected-components A connected-component is a complete section of connected ink trace, delimited only by where the writer has lifted the pen. The first stage in extracting graphemes is typically to lift whole connected-components entirely from the input page. This can be done in several ways, and is sometimes combined with text-line extraction. Potential issues include handling components that connect across text lines, and components that merge with non-text page elements.

Grapheme extraction A grapheme is a character-level segment of the ink trace that may contain more or less than a single alphabetic character. Graphemes are generally used as replacements for characters in any situation where the precise content of the character is not important, as they are far easier to extract automatically. The term fraglet is sometimes used; the terms are interchangeable. Graphemes are derived by splitting the connected components in the writing direction as necessary, using a suitable deterministic heuristic. Figure 3.5 demonstrates the ink-trace minima heuristic typically used.

[1] See Section 2.2 for further details

FIGURE 3.5: Example grapheme splitting points by the minima heuristic, as given in Bulacu and Schomaker (2007a)

3.3.2 Processing Implementations

In this section, the relevant implementation details of the image preprocessing are described, along with the tools and libraries used.

Cleaning and Binarisation ImageMagick[2] is a specialised image processing library which provides extensive thresholding, despeckling, filtering, and de-noising functions. These were accessed via a freely available wrapper library for Java, JMagick[3]. Experimentation led to the conclusion that a median filter was most effective, and that applying despeckling beforehand sometimes improved the end result.

The median filter examines a window of a specified radius r around a pixel, and replaces it with the median values of all pixels in the window. In a binary image, this equates to using whichever value in the window (ink or background) has the majority. As $O(r^2)$ pixels need to be examined for each input pixel, large radii are rarely used.

Besides de-noising, the median filter has the side-effect of slightly enlarging the ink trace and filling in small gaps, and removing very fine strokes. These modifications were deemed acceptable as the dilations are fairly modest and the gap-

[2] http://www.imagemagick.org
[3] http://www.jmagick.org

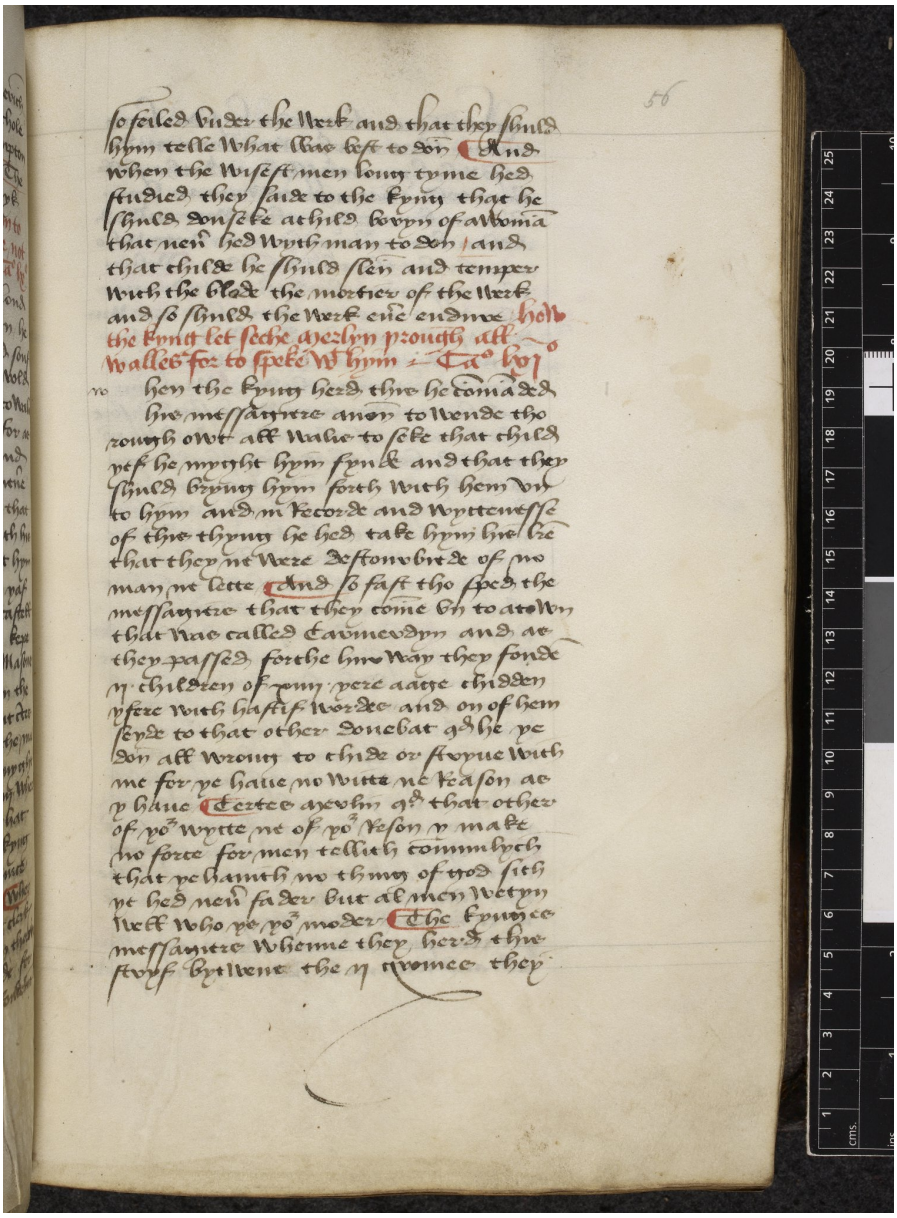


FIGURE 3.6: Sample original scribe image, ©British Library

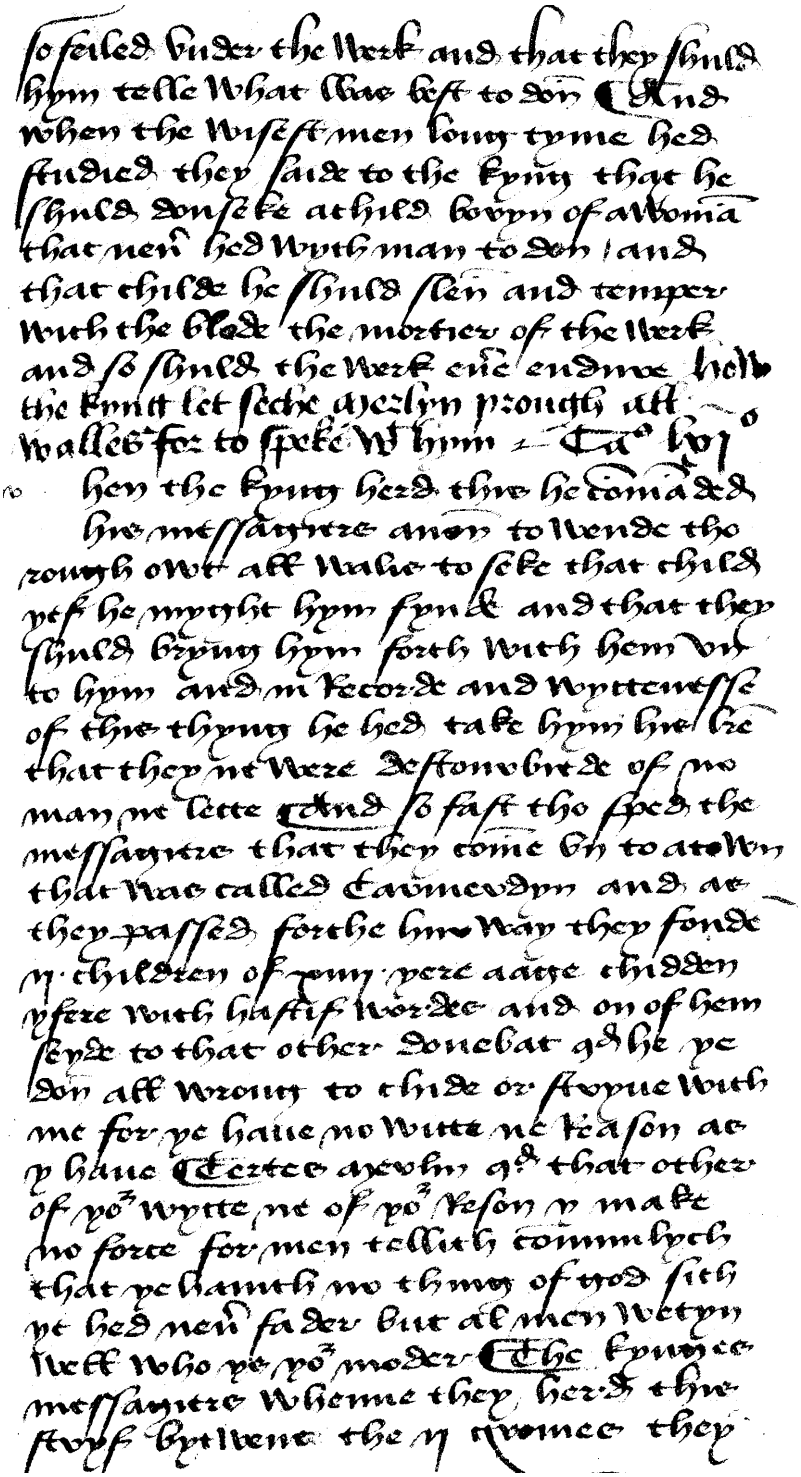


FIGURE 3.7: Sample scribe image after cropping and thresholding

filling is usually beneficial (as it most commonly covers ink trace degradation). The fragmentation of the ink trace that occurs from stroke loss is not necessarily a problem, as it is consistently applied. Although a few strokes will be lost in the process, it is an acceptable trade-off: a radius of two (that is, a 3×3 pixel window) was found to offer the best compromise between clean up and ink trace alteration.

A GUI was developed to cover the interface to the three selected functions (thresholding, median filter and despeckling), as there was no existing program that had these available to use in a streamlined fashion. The alternative was the command line interface to the ImageMagick library, but this offers no immediate visible feedback, requiring a separate image viewer to be run each time a change is made. The custom GUI allowed thresholding to be accurately adjusted, and median and despeckling filters to be tested in combination to efficiently clean the images.

Connected-component and Grapheme Extraction The extraction of the connected components and graphemes is a fully automated process. The input expected is the cleaned and binarised sample image, and the output is PNG-encoded images, which are sorted either by size (into accept/reject) and by a rough measure of complexity: the number of horizontal ink runs across the centre of the image, or maximum number of runs across the grapheme.

Connected-component extraction starts with all ink pixels from the input image, and merges adjacent groups agglomeratively. The simple algorithm developed is given in Algorithm 1 in Appendix C. The list of known components is initialised to empty, and each input pixel is examined in turn. All known connected-components that are adjacent to the current pixel are collected. If there is one or more, they are all merged together, along with the current pixel. If the current pixel does not touch any known component, it is added to the list as a new component.

After a single pass through the document input pixels, all connected-components

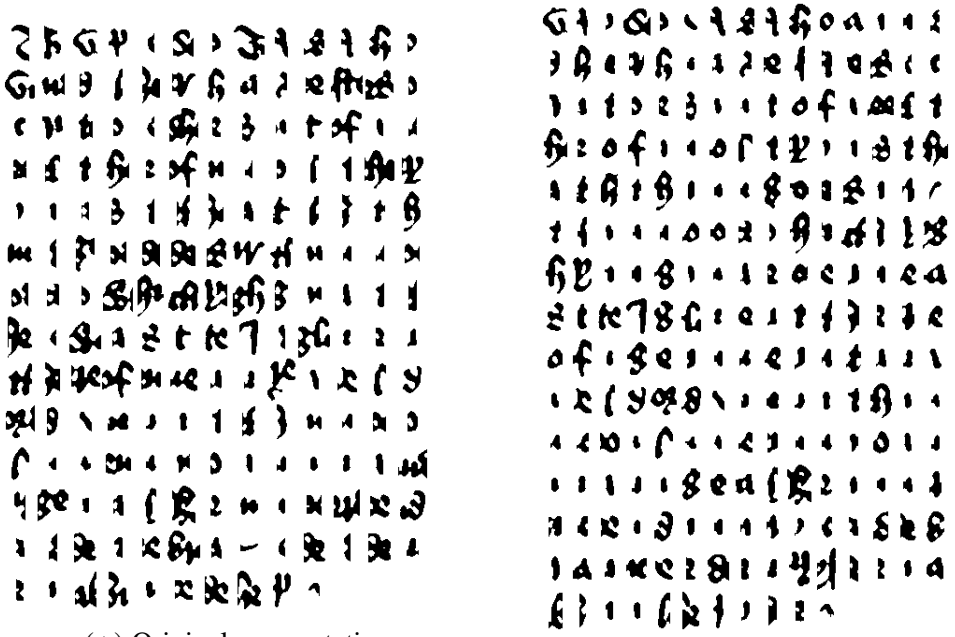
will have been extracted, however, the complexity noticed in practice is related to the connectedness and size of the components, which varies from image to image.

Graphemes are split from the connected-components in a modified implementation of the algorithm described in Bulacu and Schomaker (2007b). The original proposed splitting on "the minima in the lower contour with the added condition that the distance to the upper contour is on the order of the ink-trace width" (pg. 708). The upper contour is constructed from the uppermost ink pixel at each vertical position in the image; likewise the lower contour is the vertical position of the lowest ink pixel in each column. Ink trace minima are the lowest inflection points in the curves formed by these contours: see Figure 3.5 for an example illustration.

Experimentation found that the quality of graphemes produced by this criterion alone was not good on this dataset, and so some modifications were made: the upper rather than lower contour is searched for minima, the ink trace before and after the split is checked to see that connectedness is maintained, and a small amount of leeway is given in choosing minima (along a flat section of ink trace, any pixel may be chosen that meets the earlier criteria). This avoids over-fragmentation, and increases the likelihood of making a good cut.

The resulting graphemes have fewer rejections due to size or complexity, and are visually neater, with fewer cuts through large or complex sections of ink. These graphemes are rescaled to a fixed 50×50 pixel square which does produce some scaling artifacts, but affects mainly the lower-resolution images.

Both connected-component and grapheme extraction have rejection thresholds (based on the median component dimensions) built in to remove tiny and irregular components (e.g. underlines, noise). These underwent iterative adjustment on a pilot dataset to obtain the current result, but generalise well over the full datasets.



(A) Original segmentation

(B) Modified segmentation

FIGURE 3.8: Graphemes produced with original and modified segmentation algorithms

3.4 Grapheme Codebook

This section will discuss the grapheme codebook process in detail, covering the feature extraction and classification stages of a biometric writer identification system, as included in the 'Feature Extraction' and 'Classification' sections of Figure 3.1.

3.4.1 Codebook Generation

Once initial processing is complete, each image in the dataset is represented in the writer identification system as an unordered bag of grapheme images. This section will describe the identification process using the grapheme codebook method, discussing existing results and implementations in the literature. The first stage is to generate a codebook: a reference set of graphemes that will be retained through-

out the process to define a 'shape alphabet' with which to describe each dataset image.

Codebook Data

Codebook generation methods vary – the graphemes may originate from the dataset in use or a separate training set, and may be selected using different criteria. Drawing from the dataset in use produces a codebook most representative of the data being tested: the range of shapes that the codebook discriminates will be most closely tuned to the shapes actually used by the writers. The disadvantage is that the codebook may be overfitted to the data used to train the writer identification system. The range of ink trace shapes in unseen test data may be noticeably different to the training data, giving less accurate results in practice than those estimated from the initial training. Another potential problem is that graphemes selected for the codebook are not removed from the image representation, leaving a handful of graphemes with artificially exact codebook matches in later comparisons. This effect is only an issue in practice if the codebook comprises a significant proportion of the dataset.

A second option is to set aside a portion of the training data for use only in generating codebooks. This overcomes the problem of exact matches, but reduces the data available for training the system. The third option is to use a different dataset for generating codebooks, e.g. Schomaker and Bulacu (2004) selects codebooks from lower-case text, but includes tests run on upper-case text; Bulacu and Schomaker (2006) uses the ImUnipen dataset for codebook generation and the Firemaker and IAM datasets for system training. This mitigates the overfitting of codebooks to training data, but can also reduce the effectiveness of the final system, as the codebook may be less sensitive to the target writing style.

Grapheme Selection

No existing work directly examines the question of composing a codebook that best discriminates between writing styles; instead, codebook selection methods aim to produce a representative sample of the available graphemes. The original selection method proposed (Bulacu, 2007) was to cluster the graphemes by shape-based similarity using a Kohonen Self-Organising Feature Map (SOFM). The number of clusters is fixed to the required codebook size, and the cluster centres are selected for the codebook as they are taken to be representative of their cluster of similar graphemes. The SOFM requires extensive training to converge on a layout that best spans the shape-space – both one- and two-dimensional variants have been tested (Bulacu and Schomaker, 2005a), although the resulting spatial organisation of graphemes is not used, as the codebook itself has no intrinsic ordering.

K-means, an alternate method of clustering graphemes, has also been tested (Bulacu and Schomaker, 2005a). The identification accuracy of codebook selected this way was indistinguishable from the far more resource-intensive SOFMs. Both clustering methods aim to represent the full span of shapes in the pool of available graphemes. In doing so, the rare and 'outlier' shapes are implicitly over-represented in a codebook compared to their natural rate of occurrence.

In contrast, van der Maaten tests the simple method of selecting graphemes at random from the pool (van der Maaten, 2005), retaining the property that grapheme shapes will, on average, be selected in proportion to their natural frequency of occurrence. Identification performance using this method appears to be broadly similar to the clustering methods, although noticeably more varied. This is expected, as clustering-based codebooks calculated from the same input pool of graphemes are likely to be far more similar in composition across runs. Another notable aspect of this work is that the codebooks generated are visually very different to those reported in Bulacu (2007). This may be due to differences in grapheme preparation, but may also reflect to some degree the difference in

selection method.

To summarise, there are two main approaches to selecting a representative codebook. Clustering methods aim to span the total shape-space, akin to maximising the variance of the shape data, with the side-effect of over-representing atypical shapes. Random selection more accurately models the true shape distribution, but potentially at the expense of being less able to characterise infrequently-occurring grapheme shapes.

3.4.2 Feature Extraction

Once the codebook has been selected, the next stage of the process is feature extraction: numerically describing each handwriting image using the codebook as a reference. This is done by comparing each grapheme extracted from the image with every grapheme in the codebook, and tallying it against the closest-matching codebook element. This forms a frequency distribution of the shapes in the input image with respect to a specific codebook. It characterises the frequency with which the different types of shape occur. The frequencies are normalised by the number of graphemes in the input image to form a probability distribution, making the distribution invariant to the amount of text in each sample. Figure 3.9 shows the feature vectors generated from two IAM dataset sample images, as compared against a codebook of three graphemes. In practice, codebooks typically contain hundreds of graphemes, as smaller sizes do not contain enough information to accurately discriminate between writer styles.

Two variations have been made from this outline: at the point of comparing graphemes, and the tallying method. Grapheme comparison methods are dependent on the representation of the grapheme images. The initial contour representation used the Euclidean distance between the stored points (Bulacu, 2007), but later variations use bitmap image representations. As the images are binary and identically-sized, a image correlation measure can be implemented by checking whether the pixel at a point (x,y) in both images has the same value. A disadvan-

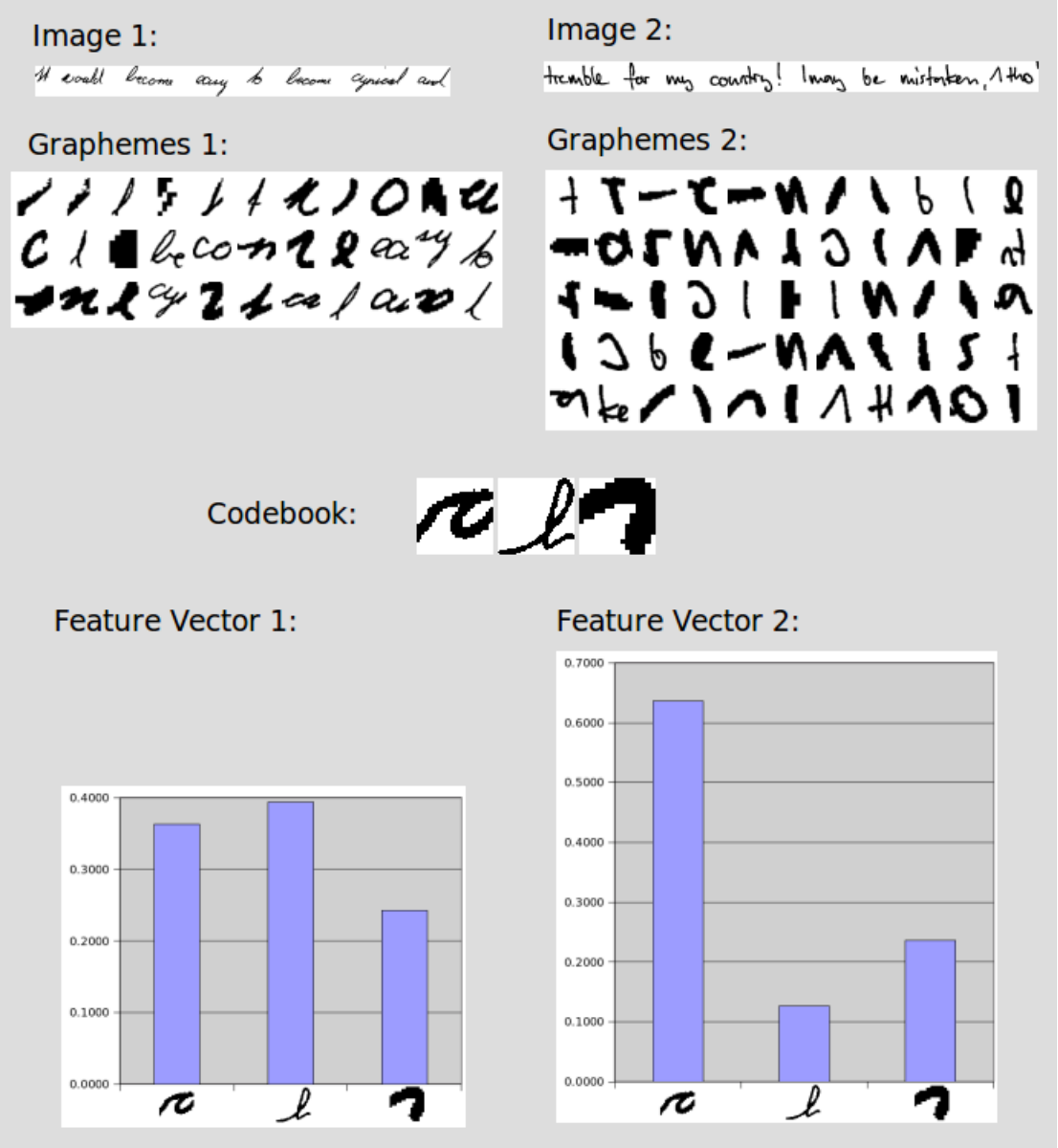


FIGURE 3.9: An example codebook of size 3, and the feature vectors generated from two IAM dataset sample images

tage of this approach is that it is not translation invariant, i.e. if the image detail in the graphemes is similar but offset, a low correlation may be registered as the pixels do not overlay (more details are given in Section 4.5). This problem can be solved by using an alternative similarity measure such as cross-correlation, which effectively measures the correlation at all possible offsets, choosing the maximum. However, it is more expensive to compute and more complex to implement (usually via the Fast-Fourier Transform). Although common in the more general bag-of-words methods for arbitrary images, it has not been tested in the codebook method literature.

Results of varying the tallying method have been reported in Schomaker et al. (2007). Instead of attributing a single tally mark to the best-matching grapheme, a count is also attributed to a number of the most similar graphemes found. The effect of the technique of 'smearing' the tally is to stabilise the assignment of increments to the codebook graphemes and smooth out the histogram, so that individual entries which match only fractionally better do not receive such a disproportionate boost in probability. This smoothing of the histogram is particularly beneficial to small samples, where edge-cases can have a relatively large effect. The number of additional graphemes incremented ranged from zero (i.e. standard tallying) to 140 in a codebook of 1089 graphemes, with a peak boost in accuracy from 75% to 81% when 30 grapheme neighbours are included.

An interesting extrapolation of this method would be to increment all codebook graphemes in proportion to their similarity to the target grapheme, but this has not been tried. This variation would also have the property of breaking the interdependence of codebook features, as the increment values a codebook grapheme receives would depend only on the absolute strength of the match with the test grapheme, rather than the comparative strength relative to the other graphemes in the codebook.

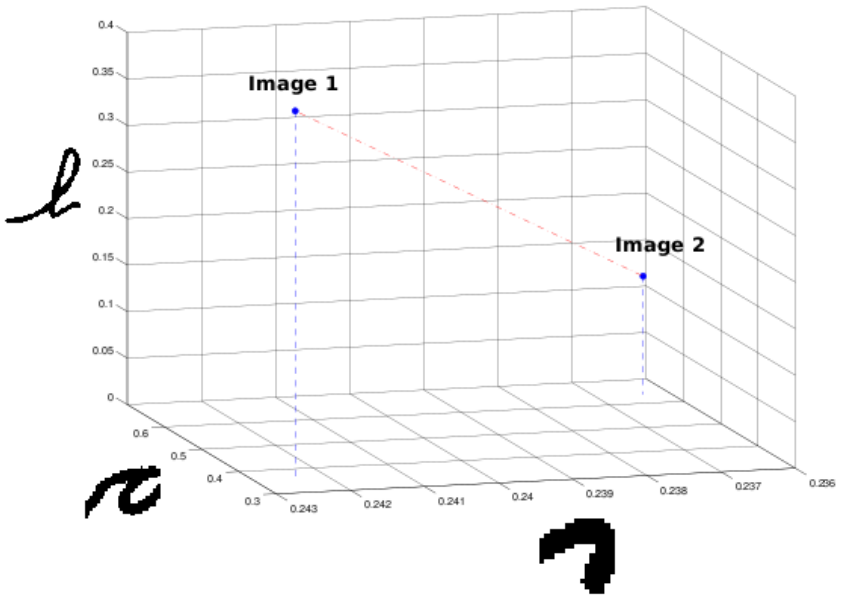


FIGURE 3.10: A plot of the feature vectors generated from the samples in Figure 3.9, in the feature space defined by the codebook. The red line indicates the distance between the samples

3.4.3 Classification

Once feature vector calculation is complete, each image will be described by a probability distribution quantised into one bin per codebook grapheme, as represented in Figure 3.9. An alternative visualisation of these feature vectors is to plot them as points in a feature space described by the codebook. Each grapheme feature in the codebook forms an axis or dimension in the feature space with values ranging from 0 to 1, and each image's feature vector can be plotted as a point using its corresponding probability values for each codebook feature. Figure 3.10 illustrates these points for the IAM samples from Figure 3.9.

To identify the writer of an image, its feature vector must be calculated and compared to the feature vectors belonging to images whose writers are already known. The simplest method is to measure the distance between the points in feature space, with proximity indicating similarity. This is the unweighted or Eu-

clidean distance metric; alternative metrics such as Hamming, Chi-square, Bhattacharya (Schomaker et al., 2004) define the distance between two points differently, weighting some dimensions more significantly than others. Schomaker et al. (2004) finds that Hamming performs best, but Schomaker and Bulacu (2004) uses the chi-square distance. The authors explain that with the latter metric, differences in the low-probability areas of the distributions are emphasised.

Having chosen a metric for comparing feature vectors, the next stage is classification: assigning a sample of unknown identity to an existing group. The most widely used method in the writer identification literature is nearest-neighbour – the unknown sample is assumed to have the same writer as its closest neighbouring sample in feature space. This classifier effectively operates by partitioning the feature space around the points belonging to each class, regardless of the shape this forms (including disjoint areas). This allows it to faithfully reflect highly irregular boundaries.

Chance-levels and Significance

To determine the usefulness of a feature or set of features, a reference point is required. The most basic comparison that can be made is whether the feature performs better than chance-level, i.e. choosing a writer class at random as the output. Given n writers, the naive chance level is $1/n$ and assumes that there is an equal chance of choosing every writer class. However in the case of nearest-neighbour, classification is based on proximity to one of the existing training-set writing samples. This assumption therefore only holds if the training set contains an equal number of samples for each writer. To calculate a more accurate chance-level in this case, the probability of choosing a given writer class must be weighted by the proportion of its training-set samples. For instance, if a training set contains four samples from writer A and one sample from writer B, the chance of a test sample being assigned to writer A will be four times that of writer B, i.e. 80% and 20% respectively. In the experiments reported in this thesis, the more accurate

per-sample method of calculating chance levels has been used throughout.

The second aspect of comparing classification results is significance. As the methods used for identification involve an element of random selection, the same experiment is run multiple times, and the average (mean) result taken to produce a more reliable estimate of the method's performance. The more runs, the more reliable the mean will be as a true performance indicator, but it is still subject to a margin of error. Additionally considering the spread of these individual results allows us to estimate the range in which the true mean identification performance lies. This statistic is the *standard error of the mean*, often shortened to standard error. The probability that the true mean lies within within one standard error above and below the calculated mean is approximately 68%; a separation between two results of at least this range is the basic requirement for the result to be *statistically significant*. If two methods give results for which the standard error ranges overlap, there are insufficient grounds on which to confidently state that either method performs better, i.e. the result is not significant. Confidence that two results are different increases with the separation between their standard error ranges. Where relevant, the error bars on plots in this thesis will be $\pm$ one standard error to illustrate this range.

3.4.4 Codebook Summary

This section has described the basic codebook method of generating a reference codebook, calculating feature vectors for all training and test images, and assigning test images the identity of the closest image's writer. The variations in method and implementation available in the literature have been discussed and compared. The next section considers these alternatives to produce a consistent methodology as a basis for future work.

3.5 Methodology

This section documents the development of a standard experimental methodology, including some preliminary experiments carried out to support the approach taken. For the most part, the most common approach taken in the field has been retained to aid comparability. Where several options are available, the simplest method was usually implemented. As this work is concerned more with comparison than extracting the maximum identification performance, this allows the effects of various method adjustments to be more easily seen.

Grapheme Extraction Binarised images were used as input to the grapheme extraction method. Some variations in implementation detail were required to cope with the image noise present in the medieval dataset, but once these were developed both datasets were processed using the same parameters. Minima-segmentation was retained as the standard method, as was aspect-ratio normalisation to 50×50 pixels for the resulting bitmap images.

Image Distance The simple image correlation distance was used to compare grapheme images. The translation-invariant cross-correlation was also tried, but computation time was prohibitive. Further details on this experiment can be found in Section 4.5.

Codebook Size Codebook sizes of 50, 100, 150, 200, 250, and 500 were chosen, as the results in Bulacu and Schomaker (2007b) suggest performance peaks at codebooks of around 100 - 400. Preliminary experiments were conducted on both datasets with codebook sizes ranging from 18 – 2500 graphemes, which confirmed the figures found in the literature.

Multiple runs As all of the experiments in this thesis involve some element of random selection, multiple runs were used for each experiment to increase the reli-

ability of the results. This was typically eight per parameter combination wherever computation resources permitted, with mean and standard error reported.

Classifier Again, the standard nearest-neighbour classifier was used, for both comparability and clarity. The Euclidean distance was used to compare feature vectors, as an unweighted comparison of the feature vector distributions was preferred. Although more than two samples from each writer are available, the leave-one-out testing strategy typically found was also retained, allowing robust results to be calculated whilst making maximum use of the training data. Writer identification accuracy will be primarily reported based on the best match only (Top-1).

3.5.1 Experimental set-up

Unless stated otherwise, the methodology for experiments is therefore summarised as follows:

- Binarised page images as input
- Approximate grapheme splitting on ink trace minima
- Storage and comparison as bitmapped 50×50 pixel images
- Codebook construction by random selection
- Simple image correlation as a grapheme difference measure
- Direct use of probability distributions as feature vectors
- Euclidean distance for comparing feature vectors
- Nearest-neighbour classifier
- Leave-one-out testing strategy
- Eight runs per parameter combination

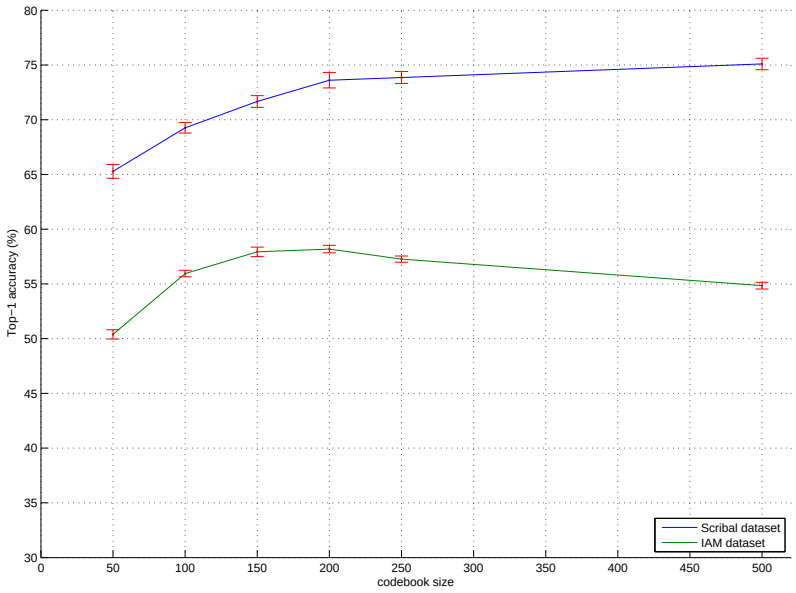


FIGURE 3.11: Baseline Top-1 identification results on the medieval dataset (43 writers) and IAM dataset (93 writers)

- Accuracy reported from Top-1 match

3.6 Baseline Results

To create a target against which to compare method variations, the baseline performance was measured for the experimental set-up described above. The plots in Figure 3.11 give the Top-1 identification accuracy on the scribal and IAM datasets. The baseline accuracy for the IAM dataset is significantly lower than the medieval data due to a larger number of writers and a much smaller sample size: the scribes dataset images contain up to a page of text (an average of approximately 1000 graphemes), whereas the IAM dataset consists of text-line samples of around 35 graphemes.

The scribes dataset accuracy increases continuously with codebook size, al-

though the rate of increase drops after 200 graphemes. The identification accuracy of the IAM dataset peaks at a codebook size of around 150–200 graphemes before dropping off on larger codebooks.

3.7 Summary

This section has documented the details of the data processing and experimental preparation made for the work of this thesis. The codebook method chosen was discussed in detail, and an experimental methodology has been defined from the outset to ensure experiment results are consistent and comparable. As briefly mentioned in Section 3.5, some variations on this method have been tested rigorously to observe whether the IAM and medieval datasets respond differently. These experiments form the next chapter.

Chapter 4

Grapheme Codebook Analysis

As described in the previous Chapter, the grapheme codebook method has been chosen for these investigations for its high identification accuracy and implicit adaptability to a range of script styles. In this Chapter, three sets of experiments are described which examine different aspects of this method.

The first two experiments (Section 4.1) focus on different ways of extracting graphemes from the original image ink trace, and the third experiment considers three methods of calculating how closely two grapheme images match.

Their primary purpose is to compare the modern and medieval datasets to discover whether each dataset responds in the same way to adjustments in processing and the codebook method, and whether some parts of the ink trace are more writer-informative than others. Checking the response of the two datasets aims to determine if current methods are directly applicable across all datasets, or whether historical data of this kind requires a different approach. Highlighting aspects of the writing that are particularly useful to writer identification enables us to see if writer-specific information is found in the same aspects of all handwriting, or whether modern or medieval texts have their own particular quirks which could benefit from closer investigation.

The experiments are reported as follows: Section 4.1 introduces the background to the first two experiments. Section 4.2 considers how much writer-

specific information grapheme aspect-ratio carries in each dataset. Section 4.3 compares methods for splitting the ink trace into graphemes. It compares graphemes that emphasise the character body with those that emphasise the between-character ligatures, and considers whether this information is complementary. Section 4.5 compares three techniques for measuring the similarity between two graphemes, including whether the position of the grapheme in the image is a significant factor in practice, and how informative the level of image detail is. Finally, Section 4.6 summarises the results and conclusions from the work in this Chapter.

4.1 Grapheme Extraction

The first two experiments involve varying the graphemes initially extracted from the image ink trace. The grapheme codebook method is considered a special case of the bag-of-words strategy for general image classification (Li and Perona, 2005). A major advantage of this specialisation is a natural and meaningful image segmentation which takes into account the writing structure. The typical segmentation method used assumes a binary input image (black text on a white background), and heuristically inserts vertical breaks at the ink trace minima. This method was originally given in Casey and Lecolinet (1996) for Optical Character Recognition and aims to produce the most character-like segments possible, but this occurs at the expense of breaking up the joins between them. Ghiasi and Safabakhsh observe that these joins, or ligatures, between characters contain writer-specific information which can be lost using standard segmentation. They propose an alternate method that combines different sizes of fixed-width segmentation, with good results (Ghiasi and Safabakhsh, 2010).

After segmentation, graphemes can be represented as contours or bitmaps, with little impact on algorithm performance (Bulacu and Schomaker, 2005a). Using bitmaps, the graphemes are usually normalised in size to a uniform 50×50 pixels, preserving the aspect ratio. However, Schlapbach and Bunke (2005)

find that some types of text size normalisation reduce identification accuracy. Fornéz et al. find that constant-size normalisation in musical notation consistently gives a higher identification accuracy than normalisation that preserves aspect-ratio (Fornéz et al., 2009).

These results suggest that the typical approach to grapheme extraction may not always be optimal. These experiments therefore test alternative methods for the two main aspects of grapheme extraction: segmentation from the cursive ink trace, and size normalisation for grapheme similarity matching. The identification accuracy of the codebook method with these modifications is tested on both the IAM and medieval datasets, to see whether they respond differently.

In Section 4.2 two grapheme size normalisation methods are tested: square-ratio and aspect-ratio. The square-ratio method scales all graphemes to fill a 50×50 pixel square, while aspect-ratio scales by only the largest dimension, preserving the original height:width ratio of the grapheme. In Section 4.3, a variable-width segmentation method is proposed which complements the minima-split approach by preserving ligatures. This is compared against the standard method, and the combination of graphemes from both methods.

4.1.1 Methodology

As these two experiments are closely linked, this section describes the experimental methodology used in both cases.

Codebook Method

The grapheme codebook method first splits the ink trace of an image into approximately character-level fragments, using some segmentation method. A reference set of graphemes is produced by selecting a subset of these – the codebook. Selection can be by Kohonen Self-organising Map (Schomaker and Bulacu, 2004), k-means clustering (Bulacu and Schomaker, 2005a), or random selection (van der Maaten, 2005); overall identification accuracy is essentially independent of selec-

tion method (van der Maaten, 2005).

The features for each image are formed by measuring the similarity of each of its graphemes to each of the codebook graphemes, and binning it against the closest match. The resulting probability distribution is the sample's feature vector; codebook size determines the dimensionality.

Experiment Methodology

The following experiments consider only the initial process of segmenting and storing the graphemes, and the effect that this has on classification accuracy. The standard method described in Section 3.5.1 is followed, with the exception of grapheme extraction as the variable under test. Codebook graphemes were selected randomly from the total pool of graphemes generated for a dataset for the given normalisation/segmentation method combination. Each experiment is run eight times with a new set of random codebooks generated for each run (for a total 384 runs across both datasets, including baseline experiments), and the mean and standard error of the Top-1 classification accuracy are reported (Figures 4.2 and 4.4).

4.2 Normalisation

Normalisation methods are specific to the representation of the graphemes, and are essential to allow comparison between writings which vary in size. This experiment considers two possible size normalisation options for grapheme bitmaps: fitting either a single dimension, or both dimensions, to 50 pixels. Figure 4.1 illustrates the horizontal (columns 1 & 2) and vertical (columns 3 & 4) stretching effect that square normalisation has over the natural aspect ratio of four graphemes from a medieval manuscript image.

The aspect-ratio of a grapheme is retained by scaling both the height and width by a single ratio. This ratio is calculated from the larger of the height or

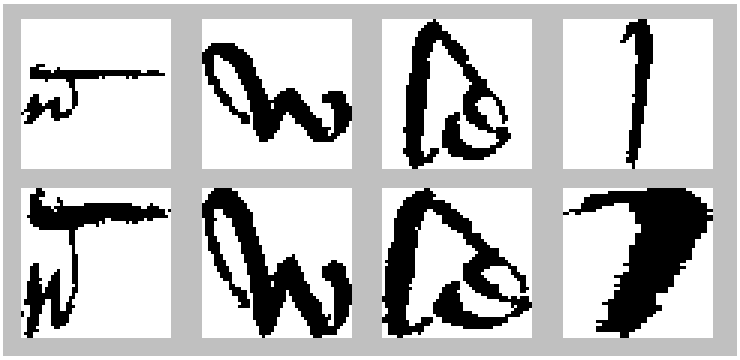


FIGURE 4.1: Comparison of graphemes produced by the ratio (top) and square (bottom) grapheme size normalisation methods

width of the original grapheme to ensure that at least one dimension of the scaled grapheme fully fits the 50 x 50 pixel frame size. Scaling both dimensions to a fixed size instead fits the frame in both dimensions, at the expense of warping the original ink trace shape to some degree. Ratio preservation retains information that may be writer-characteristic, and is the standard for bitmap normalisation in grapheme codebook experiments (e.g. stated in Bulacu (2007), by inspection in (Bulacu and Schomaker, 2007a; van der Maaten, 2005)). However, some forms of constant-size scaling in both dimensions has been shown to be beneficial to the writer identification rate. In Schlapbach and Bunke (2005), various normalisation operations were tested individually and in combination. Scaling each of the the ascenders, descenders, and centre-line regions to a fixed vertical height was the top-performing normalisation, improving identification accuracy from 76.0% (no normalisations applied) to 97.7% (only vertical scaling applied). In Fornéz et al. (2009), fixed-size normalisation for musical notation provides the highest writer-identification accuracy across all three feature extraction options tested.

In this experiment, the standard minima segmentation was used to generate two sets of graphemes for each of the scribes and IAM datasets, one aspect-scaled and the other square-scaled. As described in Section 4.1.1, reference codebooks were generated for each run by randomly drawing from the graphemes generated from the relevant dataset/normalisation combination only. The feature vector gen-

eration and classification was identical across experiments in all other respects.

Results

Figures 4.2a and 4.2b show the variation in Top-1 classification accuracy on each dataset, with error bars of $\pm$ 1 standard error (plotted with some horizontal jitter for clarity).

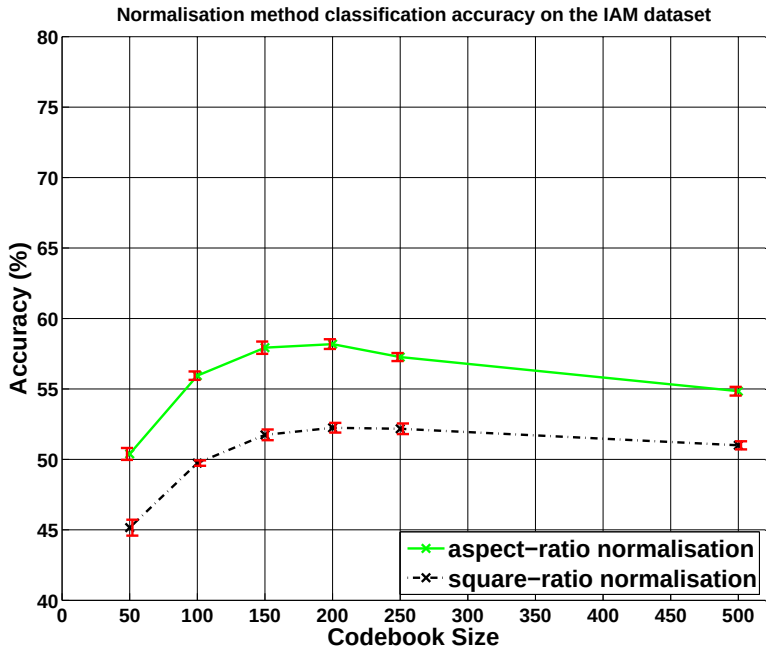
The normalisation experiments on the IAM dataset clearly show that aspect-ratio preservation performs better than fixed-dimension scaling. It produces a highly significant improvement in identification accuracy of 5–6 percentage-points, a substantial effect size equivalent to a boost of 7–11% over the square-scaled accuracy. This effect is fairly constant across all codebook sizes, and confirms that aspect-ratio in freehand Latin scripts carries writer-specific information.

On the medieval scribes dataset, aspect-ratio does not perform significantly better than square-ratio. If anything, the results trend suggests that square normalisation may confer a small (1–2 percentage-point) boost. This demonstrates that aspect ratio does not convey writer-specific information in this dataset. A likely reason for this may be found in the manuscript style rather than writer-specific handwriting style.

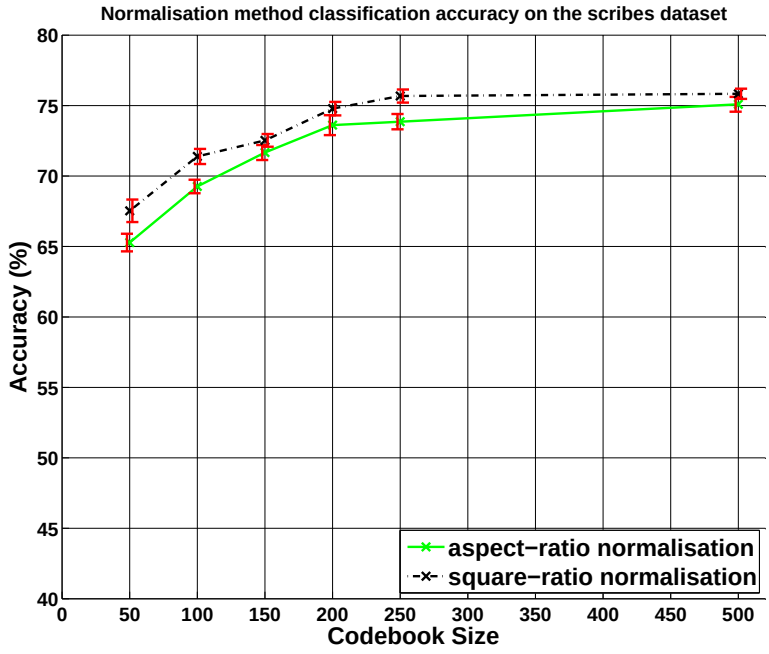
Scribes did not typically write in a personal freehand style, but adopted fonts appropriate to the manuscript. These fonts are typical of particular periods and geographic areas, and have a largely fixed aspect-ratio. This implies that aspect is likely to be more strongly correlated with font than with writer in the scribes dataset, as it is limited to scripts produced during the medieval period in England.

4.3 Segmentation

The second experiment compares segmentation heuristics, which determine how the cursive ink trace is split into usable fragments. The standard method of split-



(A) Normalisation results on the IAM dataset



(B) Normalisation results on the medieval dataset

FIGURE 4.2: Results of the normalisation experiments, mean of 8 runs $\pm$ 1 standard error

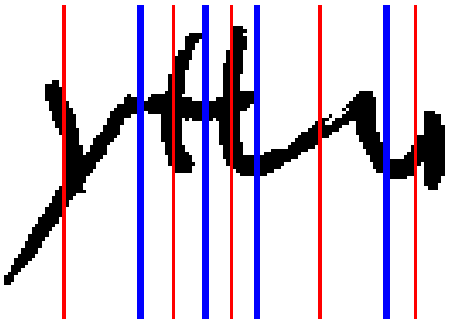


FIGURE 4.3: Comparison of splitting points produced by the minima (bold/blue line) and ligature (light/red line) segmentation methods

ting on ink trace minima (lowest inflection points[1]) aims to approximately divide it into characters, but as the focus here is the shape distribution generated by the writer and not the semantic content of the text, recognisable characters are not essential.

In this experiment, the minima method is compared with its complement, which breaks in the centre of characters wherever possible in order to preserve the ligatures instead. Figure 4.3 shows the difference in splitting points on a connected section of ink trace for each of these methods.

The minima method has been implemented by inserting a vertical break through the minimum inflection points on the lower contour of the ink trace, if it additionally holds that:

- The ink trace height at that point is approximately one stroke-width
- The segmentation will produce a grapheme with a sensible minimum width (set at 5 pixels)

The stroke-width is estimated automatically per-document from the vertical and horizontal run-length distributions.

The assumption implicit in the minima splitting method is that the character body contains the writer-specific information. An alternative hypothesis is that

[1]See Figure 3.5 and Section 3.3.2 for details of this process and an example of minima-based splitting.

the between-character ligatures contain writer-specific information, and should be preserved.

The implementation of the ligature method initially employs the same minima detection process, but splits instead at the midpoint between adjacent minima (and the connected-component boundaries where necessary). A notable effect of this process is that graphemes are no longer guaranteed to be connected-components themselves.

As these segmentation techniques are designed to be complementary, their combination is also tested. To do this, each image in the dataset is represented by the union of the bags of graphemes output by both methods: the raw image data is essentially duplicated, but each copy emphasises a different characteristic. Graphemes identical under both methods are included only once to avoid skewing the feature vector distributions in favour of single characters and small connected-components.

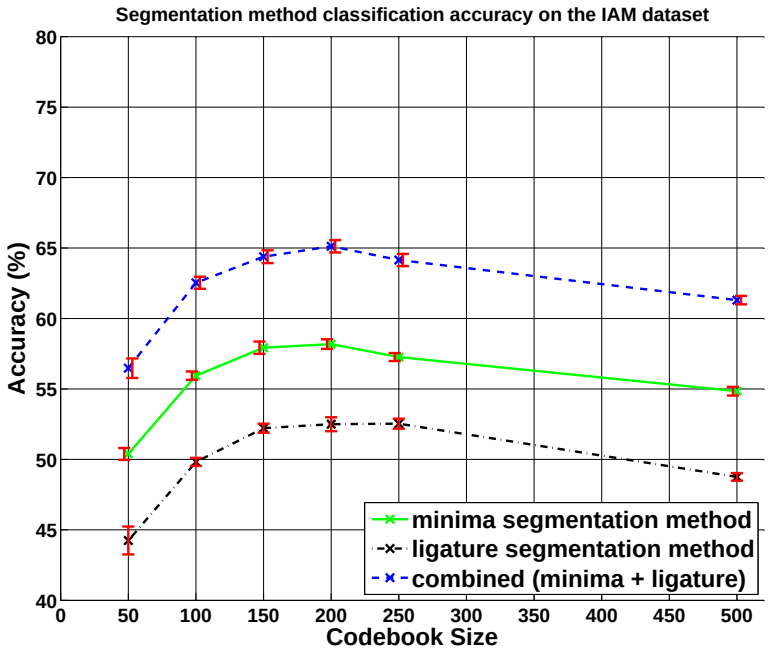
This test distinguishes between the cases where the two splitting methods produce redundant or complementary information: if there is an exact overlap in the information provided, the classification accuracy of the combination should approximately equal whichever single method (minima or ligature) is best. If the two methods are extracting different information, combining them should give a classification accuracy greater than either method individually.

Results

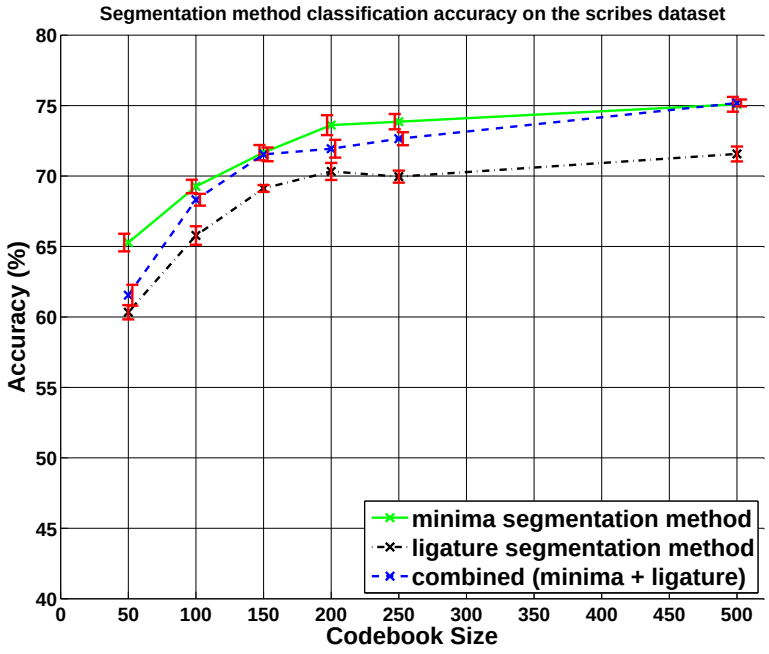
As before, Figures 4.4a and 4.4b show the Top-1 classification accuracy on each dataset, with error bars of $\pm$ 1 standard error (plotted with some horizontal jitter for clarity).

The segmentation results show that graphemes constructed preserving character ligatures do provide substantial writer-specific information, but the minima segmentation method performs significantly better on both datasets.

On the IAM dataset, combining the output of both methods gives a significant



(A) Segmentation results on the IAM dataset



(B) Segmentation results on the medieval dataset

FIGURE 4.4: Results of the segmentation experiments, mean of 8 runs $\pm$ 1 standard error

performance boost, suggesting that the writer information extracted from character body and ligatures is independent to some degree. Identification accuracy for the combined methods increases by 5–6 percentage-points over the minima-segmentation method, and by 12–13 percentage-points over the ligature method. This reflects a substantial proportional accuracy increase of 12% and 25% respectively.

On the scribes dataset, the minima-split method significantly increases accuracy by 3–4 percentage-points, or 5–7% compared to the ligature method. This confirms that the body of the character preserves more writer-specific information than a focus on the between-character ligatures. However in contrast to the IAM dataset, the combined method does not perform significantly differently to either single-strategy approach. This may be due to the much larger number of graphemes already available per-image.

4.4 Grapheme Extraction Conclusions

These experiments have examined bitmap normalisation and segmentation methods for grapheme codebooks on two very different datasets. Preserving the aspect-ratio of freehand text was found to significantly improve classification accuracy by 7–11% compared to a grapheme size normalisation that discards this information. These results suggest that at the grapheme level, aspect ratio is a writer-specific feature in contemporary freehand writing. However, likely due to the geographic and period influences of font on historical manuscripts, this does not necessarily hold true of historical data: there is at best no increase in performance from aspect-ratio preservation.

In grapheme segmentation for both datasets, preserving solely the character body provides significantly more writer-specific information than preserving solely the between-character ligatures. This effect is greatest on the IAM dataset, with a performance difference of approximately 10%, compared to a difference of

approximately 6% for the historical data. Combining multiple splitting methods produces a significant boost in accuracy on the small, clean IAM samples, but this overhead of increasing the graphemes per sample offers no advantage on the historical dataset.

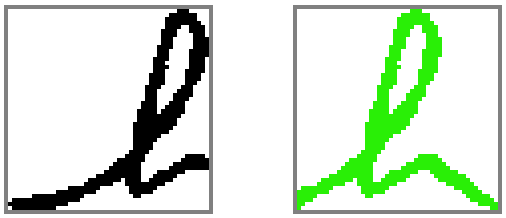
Overall, the standard minima segmentation and aspect-ratio normalisation methods appear to perform well on clean benchmark datasets, but an improvement in identification accuracy can be made for small image samples by combining multiple segmentation methods. However on the historical data, the standard aspect-ratio normalisation may have a negative impact, and combining segmentation methods offers no improvement. Extraction methods appropriate for modern freehand benchmark datasets may therefore not be optimal when applied directly to the increasing numbers of historical datasets in this area.

4.5 Image Distance measure

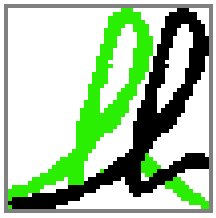
The previous experiments aimed to discover if writer-specific information is found in various aspects of the ink trace. The final experiment looks directly at comparing the graphemes once they have been generated.

The typical method for grapheme bitmap comparison is image correlation, as described in Section 3.4.2. In binary images, this is equivalent to taking the Euclidean distance between each pair of pixels, and is sometimes described as such (e.g. (Bulacu, 2007)). To implement this measure, images were represented as 50×50 matrices of boolean values, with zero representing a white pixel and one a black pixel. Correlation was implemented by taking the sum of absolute differences, normalised by the combined size of the images, and scaled to the range [0,1], where zero means the images are totally different and one means they are identical. The MATLAB implementation used is given in Listing 2 in Appendix C.

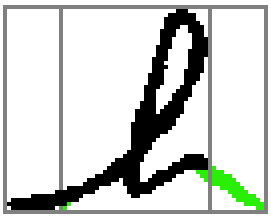
The correlation measure is good at quantifying similarities in shape, but these



(A) Two example graphemes with matching ascender loops



(B) Graphemes overlaid directly: simple image correlation finds low similarity



(C) Graphemes overlaid at best-matching offset: cross-correlation finds high similarity

FIGURE 4.5: A comparison of simple correlation and cross-correlation matching of two grapheme images

can be missed if the shapes are offset, as the ink pixels no longer line up (Figure 4.5). A translation-invariant distance measure compensates for these offsets. Cross-correlation is used here: it calculates the correlation between two images at every possible offset (padding an image if necessary) and uses the maximum value, i.e. the closest match. It also returns the (x, y) location at which the best match or matches are found, and for this reason it is commonly used to search for a small patch or template in a larger image (Briechle and Hanebeck, 2001).

This process obviously requires much more computation than simple correlation, as the similarity must be calculated for every pixel in the image. Cross-

correlation is therefore usually implemented using the Fast-Fourier Transform, which considers the image data as a signal and maps both images into the frequency domain, allowing all offsets to be considered simultaneously (Gentleman and Sande, 1966). The MATLAB implementation of cross-correlation is used here (the normxcorr2() function in the Image Processing Toolbox, Version 7.1). Values are rescaled to the range [0,1], where again zero represents total difference and one represents identical images.

As a contrast, these two correlation-based measures are compared to a distance which does not directly consider shape information. The complexity of an image is a rough measure of the level of detail it contains. This was implemented using the measure defined in Kawaguchi (2005) for binary images. It considers the proportion of black to white transitions which occur when looking along the rows and columns of bitmap image pixels, specified by

$$\frac{\sum_{x=1}^{w}\sum_{y=1}^{h-1}\|I(x,y)-I(x,y+1)\|+\sum_{x=1}^{w-1}\sum_{y=1}^{h}\|I(x,y)-I(x+1,y)\|}{w(h-1)+h(w-1)} \tag{4.1}$$

where I is the bitmap image, w is the image width, h is the image height, and (x, y) is the corresponding pixel value in the range [0,1] from white to black. If fractional pixel transitions are permitted, this formulation extends to grayscale images without modification. The MATLAB implementation of this metric is optimised by summing the changes across rows and columns simultaneously, and is given in Listing 3 in Appendix C.

Objective

The objective of this set of experiments is therefore to look more directly at how the standard grapheme codebook methodology affects both datasets. The correlation distances were chosen to examine the extent to which the graphemes formed from the standard processing of each dataset respond to a translation-invariant version of the usual shape-based correlation. The image complexity metric was

chosen as it does not directly involve shape-based information, providing instead a measure of how detailed an image is.

The research questions considered in these experiments are:

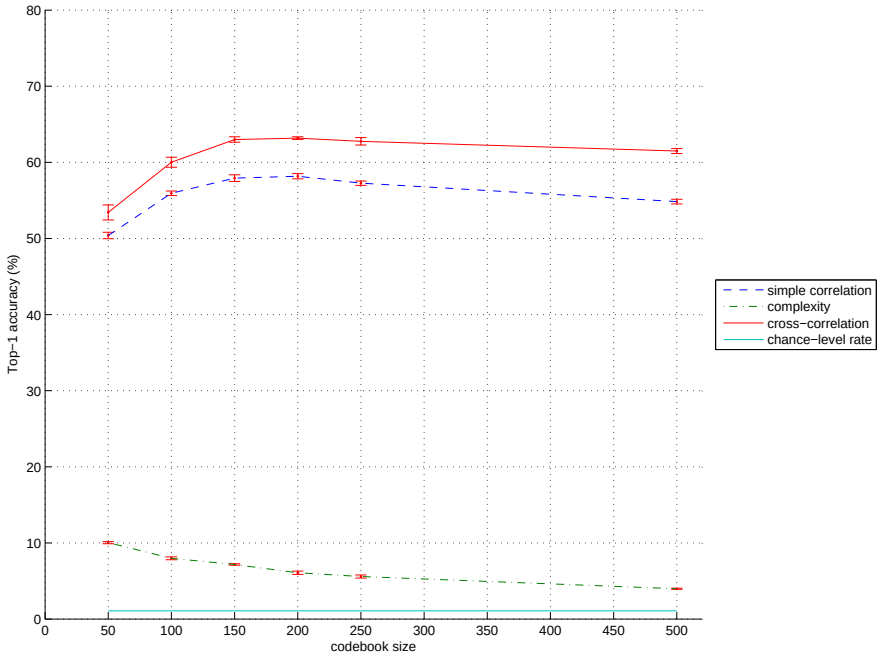
- To what extent can translation-invariance improve identification accuracy in shape-based image comparisons?
- Does translation-invariance affect the modern and medieval datasets differently?
- How useful are non-shape-based grapheme comparisons?

Methodology

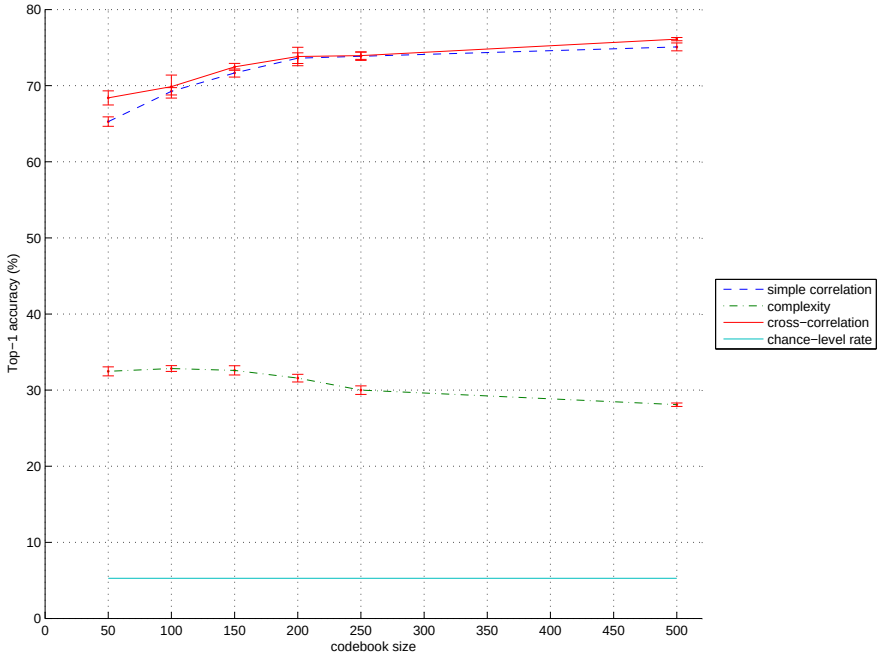
The methodology for these experiments closely follows the standard method given in Section 3.5.1, with the following notable exceptions. (The grapheme image comparison method is different in each experiment, as this is the variable under test.)

Eight runs were made for each of the simple correlation and image complexity distances. However, preliminary tests showed that the cross-correlation distance had a much higher computation time (approximately 4 times longer). Due to resource constraints, only four runs were made for this experiment, which may be reflected in a higher standard error in the cross-correlation results.

Particularly in view of the limited number of cross-correlation runs, the same set of randomly-generated codebooks was used for each distance metric, to eliminate one of the sources of relative variation between experiments. Of the eight codebook sets produced, four were used once for each of the distance metrics, and the remaining four were used for the additional runs of the simple correlation and complexity experiments. The standard set of codebook sizes was used, i.e. 50, 100, 150, 200, 250, and 500, for a total of 240 individual experiments run across both datasets.



(A) Image distance results on the IAM dataset



(B) Image distance results on the medieval dataset

FIGURE 4.6: Identification accuracy for each image distance, mean $\pm$ 1 standard error

Results and Conclusions

Figures 4.6a and 4.6b show the mean Top-1 identification accuracy of each distance metric, with error bars of $\pm$ 1 standard error. The chance-level classification accuracy is also shown for reference.

Both the correlation distances perform similarly, with the complexity distance low but substantially above chance level. The IAM and medieval datasets respond differently to the translation-invariance comparison: the IAM dataset identification accuracy is improved by approximately 2–5 percentage-points with cross-correlation, peaking at a codebook size of 150, whereas there is no difference for the scribes data. This suggests that in practice, the additional computation required for translation-invariance offers no advantage in medieval writing. A potential cause of this is the greater regularity of the scripts, leading to more consistent grapheme generation. The accuracy increase in modern handwriting may be a useful effect in e.g. forensic analysis systems, which look to extract the maximum possible accuracy, but only where the increased computation time is not a limiting factor.

The complexity distance shows a distinct decline in accuracy with increasing codebook size. The simplicity of this measure probably suffers from having too many reference codebook graphemes with similar complexity values, leading to fragmentation of the values and obscuring the underlying distribution. As performance is significantly higher than random guessing (particularly on the medieval dataset) the distribution of image detail levels is clearly a writer-specific feature – however, for practical levels of identification accuracy, shape-based grapheme comparison is still required.

4.6 Conclusions

This Chapter has presented experiments in the segmentation, normalisation, and comparison of graphemes, with the objective of comparing modern and medieval

	IAM	scribes	Section Index
Aspect/Square Ratio	Aspect	No difference	4.2
Splitting Method	Character	Character	4.3
Combined Splitting	Complementary	Redundant	4.3
Correlation Type	Cross-correlation	No difference	4.5
Image Complexity	Significant	Significant	4.1

FIGURE 4.7: Results overview for IAM and scribes datasets

datasets and the handwriting aspects that provide writer-specific information.

For the initial part of this aim, Table 4.7 summarises the results of the experiments carried out. It is clear that the datasets respond very differently overall, with both method and implementation issues to take into account when working with historical rather than modern data.

These experiments have also produced several conclusions regarding writer-distinctive information in aspects of the ink trace. The normalisation experiment in Section 4.2 demonstrated that grapheme aspect-ratio is uninformative in historical data, and may even be slightly detrimental. This result runs counter to the standard methodology in use on modern data, for which it is well suited. The difference may be due to the dominance of the imposed, non-personal font style in determining the aspect-ratio of characters.

The segmentation experiments in Section 4.3 provide three results. First, a focus on preserving the cursive ligatures rather than character bodies in segmenting graphemes still provides significant writer-specific information. Second, the ligature-based information is not as powerful at describing writers as the character-focused segmentation. Finally, these experiments showed that in modern handwriting with small image samples, these two sources of writer-specific information are complementary, as identification using the combined output of both segmentation methods was significantly better than either individually. This factor may be exploited to enhance identification accuracy when only small samples are available.

In the medieval dataset, this result was not repeated, with the combined-output graphemes failing to improve accuracy over the standard segmentation. This implies that the information relied upon by the ligature-focused graphemes is already present in the character-focused grapheme set.

In testing methods of comparing grapheme images, it was found that the intricacy of the graphemes produced gives some writer information, particularly in medieval writing. However, shape-based comparisons were required to reach accuracy levels comparable to the current state-of-the-art. These experiments also produced the interesting result that translation-invariant matching of the grapheme shapes (Figure 4.5) boosts the identification rate on the modern IAM dataset, but fails to have an effect on the historical dataset. Although less amenable to direct interpretation than earlier outcomes, this may be another side-effect of the font style used: the greater regularity noticeable in medieval scripts may produce graphemes with a more consistent layout, whereas modern or natural handwriting shapes are spatially spread over the grapheme image.

These experiments have provided significant pointers in the handling of medieval and modern datasets, particularly in analysing where writer-distinctive information may be found. The following chapter continues this information analysis, examining the feature distributions produced in the next stage of the codebook process.

Chapter 5

Feature selection and extraction

This chapter investigates several approaches to analysing grapheme codebook features for offline writer identification in medieval English scribal manuscripts. Current methods for selecting a codebook typically produce codebooks that perform no better than random grapheme selection, so the aim in this analysis is to identify potential methods of improving codebook selection. In particular, the work investigates the characteristics of a high-performing codebook. Three feature extraction methods are tested, and a number of feature selection methods are proposed and compared. In feature extraction, the values of existing features are combined into new features in various proportions, with the aim of optimising identification performance and reducing the number of new features required. A disadvantage of this approach is that any significant meaning of the original features is usually lost. Feature selection chooses a subset of the original features, avoiding this problem.

The results given in Section 5.3 show that the Principal Component Analysis extraction method performs best overall, typically matching or exceeding baseline identification accuracy on both datasets, whilst substantially reducing the number of features required. PCA-based feature selection was the top-performer of the selection methods tested, again with a significant reduction in the required codebook size while retaining good performance. The feature selection results also demonstrate that a range of grapheme-shape similarities within a codebook is necessary

for good performance. All methods are compared on the IAM and scribal datasets; the results are robust to data variation.

The remainder of the chapter is organised as follows: a description of the grapheme codebook method and existing selection methods is given in Section 5.1. Section 5.2 considers the motivation for applying feature selection and extraction to grapheme codebooks in particular, and Sections 5.3 and 5.4 describe the experiments carried out. Section 5.5 summarises the work and draws conclusions.

5.1 Codebook Selection Methods

No existing work directly examines the question of the codebook characteristics that best discriminate between writing styles; instead, selection methods aim to produce a representative sample of the available graphemes. The original selection method proposed (Bulacu, 2007) was to cluster all the graphemes by shape-based similarity using a Kohonen Self-Organising Feature Map (SOFM). The number of clusters is fixed to the required codebook size, and the cluster centres are selected for the codebook as they are taken to be representative of their cluster of similar graphemes. K-means, an alternate method of clustering graphemes, has also been tested (Bulacu and Schomaker, 2005a). The identification accuracy of codebooks selected this way was indistinguishable from the far more resource-intensive SOFMs. Both clustering methods aim to represent the full span of shapes in the pool of available graphemes. In doing so, the rare and 'outlier' shapes are implicitly over-represented in a codebook compared to their natural rate of occurrence, as cluster size is not taken into account.

In contrast, van der Maaten (2005) tests the simple method of selecting graphemes at random from the total pool, retaining the property that grapheme shapes will on average be selected in proportion to their natural frequency of occurrence. Identification performance using this method appears to be broadly similar to the clustering methods, although noticeably more varied. This is expected, as clustering-

based codebooks calculated from the same input pool of graphemes are likely to be far more similar in composition across runs.

In short, clustering methods aim to represent the dataset by spanning the total shape-space (akin to maximising the variance of the shape data) with the side-effect of over-representing atypical shapes. Random selection more accurately models the true shape distribution, but potentially at the expense of being less able to characterise infrequently-occurring grapheme shapes.

Writer invariants is a similar bag-of-words method proposed in an information retrieval context (Bensefia et al., 2002). Here, a form of repeated stochastic clustering was additionally applied to the grapheme bags for each sample to compress the training set. Graphemes which were clustered together on every run of the clustering were grouped into 'invariants'; all other graphemes were discarded. This caused no loss of precision in retrieval (98% from 88 writers) compared with using the full bag of graphemes for each sample. This result suggests that writer-specific information may be concentrated in the typical rather than the atypical shapes produced.

After codebook selection, the features for each image sample are formed by measuring the similarity of each of its graphemes to each of the codebook graphemes, and binning it against the closest match. The resulting probability distribution is the sample's feature vector, thus codebook size determines the dimensionality. To classify an unknown sample, its graphemes are again counted against the same codebook to form a probability distribution, and it is assigned the writer label of the closest-matching sample (i.e. nearest-neighbour classification). Codebook methods therefore offer a potential link between styles of character fragments and the calculated features, making it an interesting target for further analysis.

5.2 Feature Analysis

The grapheme codebook method typically produces large feature vectors: typical values can be 200-400 (Bulacu and Schomaker, 2007b), with values ranging into the thousands (Schomaker et al., 2004). Large numbers of features are usually undesirable, especially where the input sample size is small. The 'curse of dimensionality' (Bellman, 1957)[1] describes the problems that arise when manipulating high-dimensional features, particularly how the number of samples required to consistently represent an area of feature space grows exponentially as the dimensionality of the space increases (Hastie et al., 2009). The nearest-neighbour classifier typically used with the grapheme codebooks offers no mitigation of this effect.

Additionally, high numbers of features increase computation time and processing requirements, and may obscure meaningful links between the grapheme representations and salient elements of writing style. Analysis of existing features can also produce a reduced feature set that performs as well as, or better than, the original features.

The aim in this work is therefore to investigate techniques for feature reduction in grapheme codebooks according to the following criteria:

- Increasing computational efficiency
- Increasing final classification accuracy
- Determining if features correspond to meaningful writing elements

In the latter aim, discovering whether such writing style aspects are consistent or significantly different between modern and historical handwritings is of particular interest.

Techniques for feature analysis fall into two broad categories: feature *extraction* where the original feature values are weighted and recombined into some

[1] Also known in this context as the Hughes effect (Hughes, 1968)

number of new features, and feature *selection*, where a subset of the original features are used without modification.

Feature selection is simplest when the original features are independent (or nearly so), as this minimises the feature combinations that require testing. Unfortunately in grapheme codebooks, the opposite is true – the value of the feature associated with a given codebook grapheme is (theoretically) dependent on all other graphemes in the codebook. This is because each grapheme in a sample is compared against all codebook graphemes before being allocated to the closest match. Removing any single grapheme from a reference codebook will redistribute its allocated sample graphemes amongst the remaining codebook graphemes in unpredictable ways.

A set of feature extraction methods are initially considered in Section 5.3, followed by grapheme feature selection in Section 5.4.

5.3 Feature Extraction

Feature extraction methods combine the existing features in various proportions to create new features, against which the samples are measured to create a new set of feature vectors. The original grapheme features can be considered as axes or dimensions which describe a feature space in which samples are located. The process of feature extraction effectively creates a new feature space with different axes, changing the basis against which samples are placed. For the simplest feature extraction, the new feature space is just a rotation of the original space. More complex methods combine scaling, inverting, or otherwise manipulating the original feature axes with the objective of creating a new feature space which will better separate the samples into their writer groupings.

Three feature extraction methods were tested: Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA). Each of these methods calculates a new set of features as a weighted

sum of the original codebook features. Each method outputs a matrix of coefficients, giving the weighting of each original feature for each new feature. Ranking coefficients are also output for PCA and LDA to indicate the relative importance of the new features: taking the top n of these reduces the dimensionality.

PCA attempts to characterise the whole dataset by retaining as much of the original feature variance as possible into the fewest dimensions; the new feature dimensions can be ranked by the proportion of variance accounted for. ICA (Hyvarinen et al., 2001) considers the original feature vectors to be a mixture of a hidden underlying set of factors. It therefore aims to derive these underlying features under the assumption that they are independent. PCA and ICA are both examples of 'blind source separation' techniques. This type of analysis does not require any writer label information, and can be performed once for the whole dataset.

In contrast, LDA uses the writer label information to generate a set of features which maximise separation between samples from different writers, and minimise the spread of samples from the same writer. New features can be ranked by the eigenvalues which give the writer class separation that each provides. As this requires correct writer label information for the training set, computation requirements for LDA are higher than those of PCA and ICA: the mapping must be recalculated for the training set of each fold of the cross-validation in order to withhold the writer label for the samples being tested each time.

In these experiments the standard experiment set up was used (see Section 3.5.1 for details), i.e. codebook sizes of 50, 100, 150, 200, 250, and 500, simple pixel-wise image correlation to generate the feature vectors and the Euclidean-distance nearest-neighbour for classification. Leave-one-out cross-validation is used for all experiments. For both datasets, each codebook size was tested 8 times with a new randomly-selected codebook drawn from that dataset's total pool and classified using each of the four methods, giving a total of 384 experiments run. The results given below show the mean $\pm$ 1 standard error.

The baseline result is simple classification of the original feature vectors. On the scribes dataset, this is 73.6% $\pm$ 0.70 (1 standard error) for eight runs on a size 200 codebook, and 58.2% $\pm$ 0.34 for the same configuration on the IAM dataset. The differences in baseline accuracy are related to the varying number of writers and sample sizes of each dataset. All feature extraction results are given as fractions of this baseline classification accuracy.

The following sections give a technical description of the extraction methods used, and Section 5.3.2 presents the results.

5.3.1 Feature extraction details

In these sections the following symbols will be used:

- number of writers: w
- number of original features: f
- number of samples: s
- original feature vector data: $\mathbf{X} : f \times s$
- number of samples in a given writer class i: s_i
- feature vector data for a writer class i: $\mathbf{X_i} : f \times s_i$

PCA

The PCA transform is found from the eigendecomposition of the covariance matrix of feature vectors, i.e.

$$cov(\mathbf{X}) = \mathbf{PDP^T} \tag{5.1}$$

where $\mathbf{P} : f \times f$ is the eigenvector matrix containing the principal components $[\mathbf{p_1}, \mathbf{p_2}, ..., \mathbf{p_f}]$. The principal component $\mathbf{p_1}$ is the direction of maximum data variance, i.e. greatest spread. The second principal component $\mathbf{p_2}$ is defined to be the axis of maximum remaining data variance, with the constraint that it must

be orthogonal to $\mathbf{p_1}$. The remaining principal components are defined likewise. The diagonal matrix $\mathbf{D}$ contains the eigenvalues associated with each component in descending order, which are the proportion of data variance the corresponding direction accounts for. Dimensionality reduction in PCA feature space is by dropping the components associated with the lowest eigenvalues, as these represent the directions of little data variation.

PCA was chosen for its dimensionality reduction properties, as the top 10% of PCA dimensions often provide a reasonable classification rate. The mapped dimensions and their weightings may also provide some insight into which graphemes or grapheme groups are most influential in writer identification.

LDA

PCA considers only the total distribution of the sample data, without taking writer information into account. The ideal mapping of samples into a new feature space would place samples from the same writer class as close together as possible, and as far apart from samples of different classes. The aim of LDA is to approximate this transform, which is modelled by maximising the ratio of between-class scatter to within-class scatter. This can be considered as a two-stage process: a 'sphering' transform on each class, followed by PCA applied to the class means to find the directions which spread them as far apart as possible. LDA assumes that the class covariances are identical – lifting this restriction gives the related Quadratic Discriminant Analysis (Hastie et al., 2009, p. 112).

Sphering is therefore approximated by applying $\mathbf{W^{-1}}$, where the mean within-class covariance matrix $\mathbf{W}$ is

$$\mathbf{W} = \frac{1}{w} \sum_{i=1}^{i=w} cov(\mathbf{X_i}) \tag{5.2}$$

The between-class scatter $\mathbf{B}$ is given by the variance of the (centred) class

means

$$\mathbf{B} = \frac{1}{w} \sum_{i=1}^{i=w} (\mu_i - \mu)(\mu_i - \mu)^T \tag{5.3}$$

where μ_i is the mean of the feature vectors for a class i and μ is the mean of $\mu_1 ... \mu_w$. The eigenvectors of $\mathbf{W^{-1}B}$ maximise the class separation, which is given by the respective eigenvalues.

LDA was applied here as it is incorporates class-specific information, allowing it to model the grapheme combinations which best discriminate between writers. Examining these may highlight individual writer quirks or trademarks. The implementation used in this work is from the Matlab Toolbox for Dimensionality Reduction[2] (van der Maaten et al., 2009).

ICA

ICA does not consider the class information, but instead seeks to represent the data by a set of features with minimal mutual information. It does this by modelling the observed samples as a mixture of an unknown set of f source factors, assumed to be statistically independent and non-Gaussian. These requirements allow the estimation of both the independent factors (the new features) and the mixing matrix of feature weights, but do not define any ordering on the derived factors.

Given the mixing model $\mathbf{X} = \mathbf{MS}$, where $\mathbf{M}$ is the mixing matrix and $\mathbf{S}$ is the matrix of independent sources, the first stage is applying the sphering transform described above which whitens $\mathbf{X}$ and and transforms $\mathbf{M}$ into an orthogonal matrix $\mathbf{M}'$. However, as any orthogonal transformation of the data remains sphered, one of a number of estimation methods is still required to calculate $\mathbf{M}'$; the FastICA implementation[3] is used here (Hyvärinen and Oja, 1997).

The ICA model of feature data as multiple observations of an underlying gen-

[2]Available at `http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html`

[3]Available at `http://research.ics.tkk.fi/ica/fastica`

eration process maps well onto the grapheme codebook's statistical approach. If source factors can be reliably extracted from the grapheme codebook data, they would offer an excellent interpretation of the writing characteristics which distinguish the writers in each dataset.

Dimensionality

Each feature extraction method is run over the baseline feature vector data, giving varying numbers of new features. ICA produces at most the same number of new features as original features, and LDA produces at most $\max(f, w-1)$ features. PCA outputs slightly fewer new features than original, but typically the top n features are used such that they cover some percentage of the original data variance, e.g. 90% or 99%. In these data, 90% was found to offer a good trade-off between feature reduction and identification accuracy; all available LDA and ICA features were used.

5.3.2 Results and Analysis

On the medieval dataset, the performance of PCA was by far the highest and most consistent, being the only method to exceed the original accuracy. Performance increases with codebook size, and performance equivalent to the baseline accuracy is reached on the scribal dataset at a codebook size 200, and on the IAM dataset at size 500. This required approximately 30% and 58% of the number of features originally used on the medieval data (Figure 5.4) and IAM data (Figure 5.2) respectively. The number of PCA features required shows a distinct decreasing trend as codebook size increased on both datasets. Apart from the smallest codebook size, LDA performed poorly on the scribal dataset. Analysis of LDA on a smaller number of features suggests this is likely to be a numerical instability at higher dimensions, possibly related to the sparseness of the feature vectors. On the IAM data, LDA performs well at smaller codebook sizes, reaching approximately 1.1 times the baseline accuracy before dropping off sharply as on the scribal data.

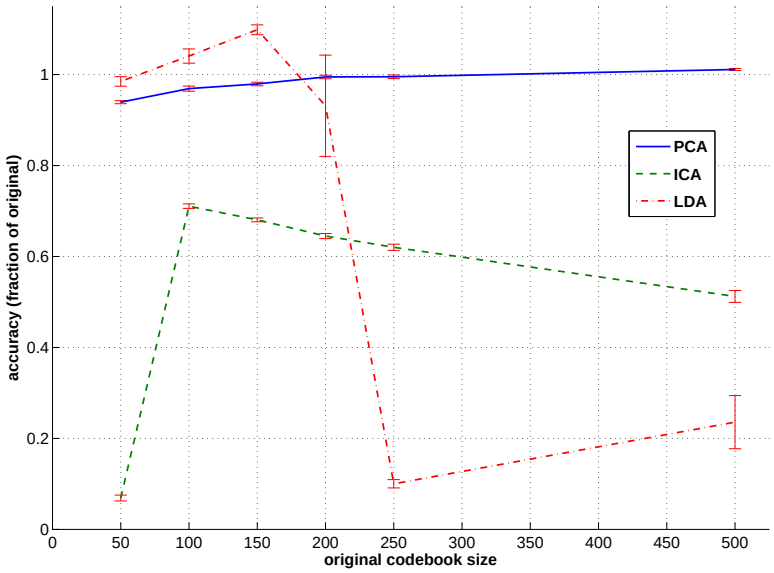


FIGURE 5.1: Classification accuracy of feature extraction methods on the IAM dataset

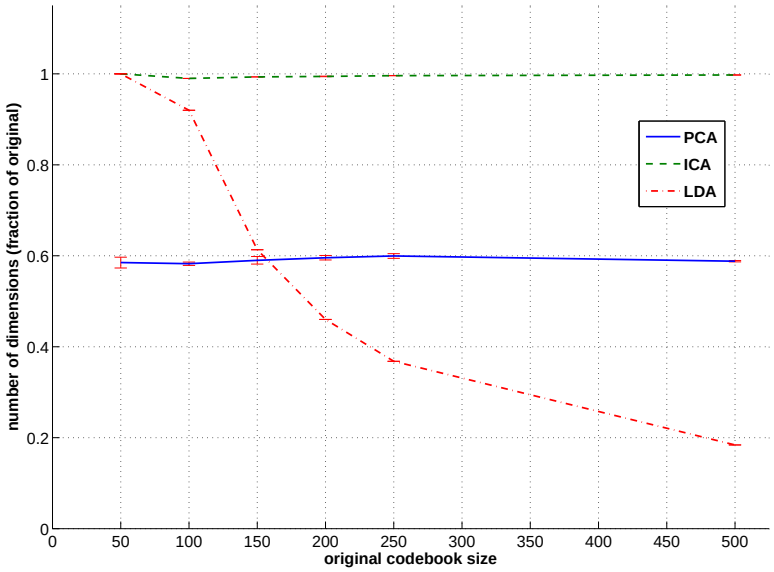


FIGURE 5.2: Number of features used in extraction methods on the IAM dataset

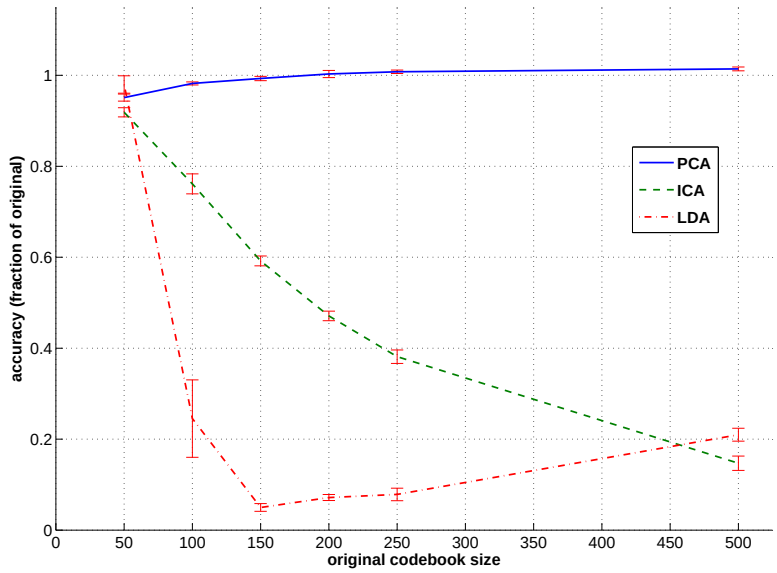


FIGURE 5.3: Classification accuracy of feature extraction methods on the scribes dataset

ICA also performs poorly, suggesting it is either unable to extract any meaningful underlying features from this dataset, or that the features it finds are only weakly writer-informative. As all available ICA features are used, little dimensionality reduction is seen. Classifying at various lower proportions of ICA feature vectors shows only a linear increase in classification accuracy – each feature contributes a roughly equal amount to the overall identification accuracy, with no single feature providing a significant contribution.

5.3.3 Visualisation

To visualise the relative importance of each grapheme in the top PCA components, the codebook images are ranked according to their weights in the PCA coefficients. That is, given the PCA transform in Equation 5.1, the weighting of a single grapheme g in a principal component c is $w_{cg} = |p_{cg}|$. Summing these

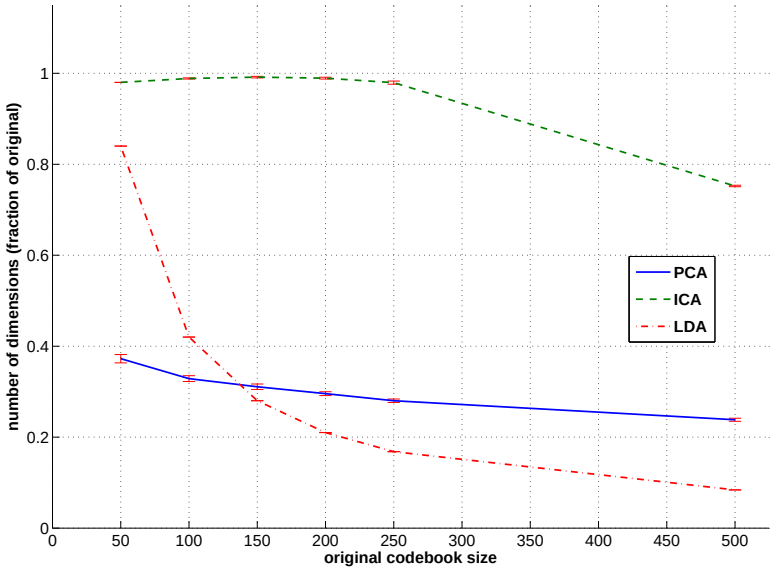


FIGURE 5.4: Number of features used in extraction methods on the scribes dataset

weights across all principal components for each grapheme produces an ordering on the codebook graphemes (Figure 5.5), most significant feature first.

Additionally, if a codebook grapheme is considered as a visual representation of a feature (or equivalently of an axis in feature space), the codebook effectively forms a visual basis of the feature space. Any vector can thus be represented by weighting the basis codebook graphemes: in particular, the individual principal component vectors can be visualised by generating the image

$$composite_c(x, y) = 255 \bullet \frac{\sum_{g=0}^{g=f} g(x, y)\, w_{cg}}{\sum_{g=0}^{g=f} w_{cg}} \tag{5.4}$$

where x and y index the image pixels. This formulation also quantises the result into 256 grayscale values. A selection of components are shown in Figure 5.6: the components along the top row are clearly dominated by a small number of graphemes, while those on the bottom row are more evenly balanced. Darker areas

FIGURE 5.5: Sample PCA-sorted codebook

show pixel positions that have greater weighting amongst the input graphemes. These visualisations can be helpful in explaining PCA and its implications for this dataset, particularly to those with no prior knowledge of classification.

However, a codebook formed of these visualised 'graphemes' cannot be used directly: the greyscale images are typically fairly balanced and thus insufficiently distinctive to form a discriminating codebook, leading to very poor performance. In order to choose a working reduced codebook, the following section considers how these coefficients may instead be used to select individual graphemes.

5.4 Feature Selection

As described in Section 5.2, feature selection in grapheme codebooks is complicated by a theoretical total dependence between features. Despite this, the ex-

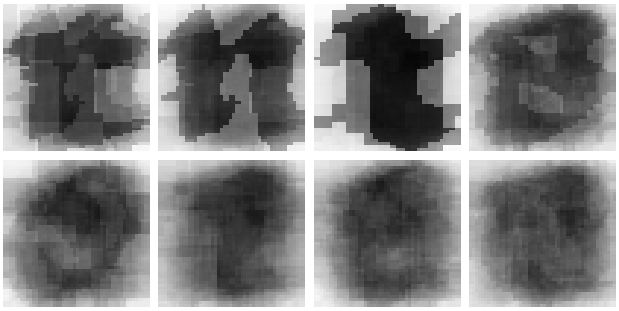


FIGURE 5.6: Visualisation of individual principal component vectors

periments in this Section first analyse codebook selection based on individually high-performing graphemes (derived from PCA and LDA coefficients) to see how well they can perform in practice.

The end goal of this analysis is to discover grapheme-image-based characteristics which can be used to improve codebook performance and better understand how to select good codebooks. The implications of this would include the ability to base grapheme selection directly on a dataset's grapheme pool, without requiring feature vector data or initial classifications (as in PCA and LDA). It would also potentially give insight into the aspects of medieval and/or modern handwriting which discriminates writers well. Section 5.4.1 describes the method used to convert feature weightings into original grapheme rankings.

Codebook selection aims to generate a set of graphemes representative of the total pool of graphemes $\mathbf{G_D}$ of all samples in the dataset. As discussed in Section 5.1 and the review in Section 2.3.5, the different clustering approaches used in the literature to date emphasise retaining the graphemes which either appear most frequently or are outliers. From the writer or scribe perspective, this amounts to looking for writer-specific information in the common allographs and characteristic strokes, or in the uncommon and distinctive graphemes. This suggests that the level of similarity of a codebook's graphemes will be an interesting image-based characteristic for investigation. Section 5.4.2 expands on this and describes the

approaches and implementations tested.

5.4.1 Extraction-based methods

For the first strategy, the feature extraction methods tested in Section 5.3 were used to calculate significant individual features. These PCA- and LDA-based feature selection methods were implemented by ordering the original features by the sum of the coefficients for each original grapheme feature, weighted by the importance of the coefficient's corresponding new feature. The top n individual graphemes are then chosen from the resulting ranking. For example, in terms of the PCA transform and coefficient given in Equation 5.1, summing over the top i principal component features used gives the following weighting for each original codebook grapheme feature:

$$w = \sum_{c=0}^{c=i} |p_{cg}| \tag{5.5}$$

This weighting describes how important a single grapheme is, given its significance in each of the new extracted features and the significance of those features in turn. Choosing a subset of graphemes in this way from a large original codebook and feature-data combination highlights the individual grapheme bins whose features were most influential in determining classification accuracy for that experiment. However, there is no guarantee that these selected features in combination form a good codebook, even with respect to the same dataset.

Due to these feature-data requirements, these experiments were set up in two stages. The baseline experiments (see Section 3.5.1 for details) were used as initial 'source' experiments: the necessary feature data was drawn from these results, and the corresponding codebook was used as the grapheme pool for selection. The baseline codebooks used are defined to be g_s graphemes in size, from which a subset of size g_t is selected according to the method being tested. Further details are given in Section 5.4.5.

5.4.2 Similarity-based methods

The second selection strategy considers a characteristic of a codebook rather than of individual graphemes: the distribution of grapheme similarities within a codebook. Similar graphemes in a codebook can provide a fine-grained distinction between near-identical character shapes, whereas the outlier graphemes in the dataset pool may highlight distinctive or unusual writer-specific characteristics.

The intra-codebook grapheme similarities are measured pairwise for all graphemes in a codebook, using the same correlation image distance used in comparison. It is therefore expected that each selection method will give a characteristic distribution of similarities, and that a potential explanation for the surprisingly strong performance of the 'random' selection method is that it covers a similar range to the clustering methods. It is further expected that selecting outliers weights the distribution towards the lower similarity range, and that selecting similar graphemes weights the overall codebook distribution towards the higher similarity end of the range.

We hypothesise that high-performing codebooks span a range of grapheme similarities, allowing both fine-grained distinctions and unusual writer-specific shapes, and that focusing exclusively on a single aspect will reduce codebook performance. In order to test this, three similarity-based approaches are examined: maximising the similarity of the graphemes, minimising their similarity, and including the widest possible range of similarities.

These three methods were implemented heuristically. To produce a codebook of mutually similar graphemes, the image distance matrix $\mathbf{D} : g \times g$ (where g is the number of graphemes in the source codebook) was calculated with the same image pixel correlation used to produce the codebook feature vectors. Summing the matrix columns gives a vector $\mathbf{t} : 1 \times g$ containing the total distance between a single grapheme and all others in the codebook. The index of $\min(\mathbf{t})$ therefore gives the grapheme which is, on average, most similar to all other codebook graphemes.

Minimal and Maximal Similarity Based on this, two implementations were tested to select codebooks of minimal similarity. The *global* implementation simply selected the *n* graphemes with minimal total distances. The *iterative* implementation made $n - 1$ consecutive selections, choosing each time the grapheme most similar to those already selected; it was initialised with the global minimum. Codebooks of maximal similarity were implemented likewise. Plotting the distance distributions of target codebooks generated by each implementation demonstrated that the global method gave better results in all cases, possibly due to the iterative method being very sensitive to the initial few grapheme shapes. The global implementation was therefore used in all the following experiments, and these selection methods were labelled sim-min and sim-max. Examples of codebooks selected using the sim-min and sim-max criteria are given in Figure 5.7. Both criteria produce codebooks with notable clusters of grapheme shapes. This is clearest in the sim-min codebooks: a few graphemes are very similar to each other, but are sufficiently different from the remaining graphemes to have a low overall similarity to the codebook as a whole.

Similarity Ranges The objective of including a range of grapheme similarities can be approached in several ways. The first option combined the sim-min and sim-max approaches: when selecting n graphemes, the top $n/2$ elements by each of the min and max criteria were taken, giving a distribution with two distinct peaks at either end of the similarity range.

The second option was to keep the full spread of similarities present in the original set by ordering all graphemes and choosing approximately equally-spaced elements such that both ends are included and n graphemes total are selected. This approach samples the existing distribution of similarities whilst retaining representation of the extremes.

The third option is to aim for a flat or even distribution of similarities by binning the graphemes into a histogram of similarities (10 bins were used) and select-

ing equal numbers from each bin (as far as is possible). As they emphasise different similarity characteristics, all three of these approaches were implemented and labelled sim-range, sim-spread, and sim-even respectively.

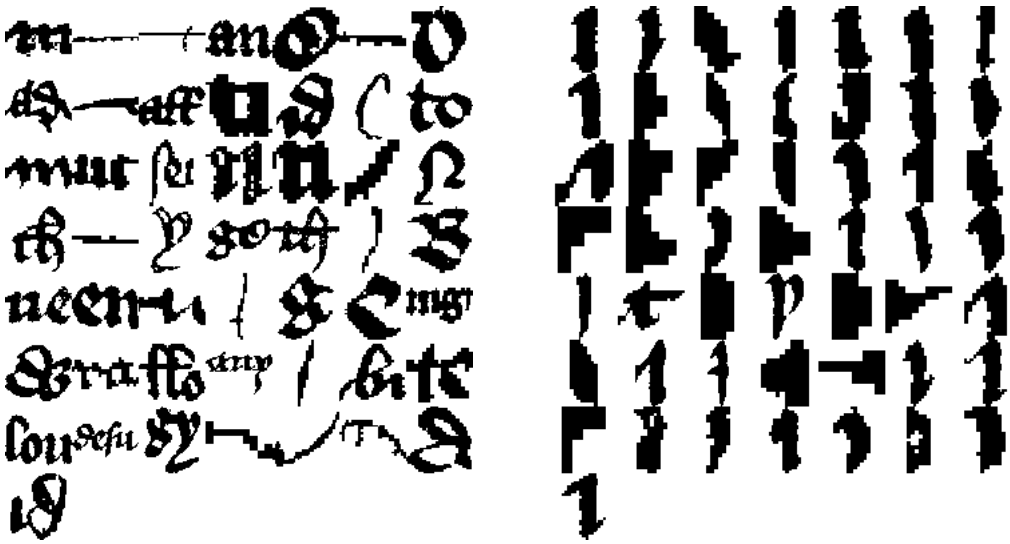
5.4.3 Hybrid method

Given the poor identification results of LDA as a feature extraction method, it is expected that it would perform poorly as a selection method as well. Preliminary analysis suggested that the similarity range of graphemes selected by LDA was narrow, and may be a contributing factor. An additional 'hybrid' strategy was therefore tested: the codebook was ordered according to similarity and used the LDA-rankings as a starting point, with the additional constraint that no two similar adjacent graphemes could be selected. This has the effect of 'spreading' the graphemes over a wider similarity range, and was labelled LDA-sparse.

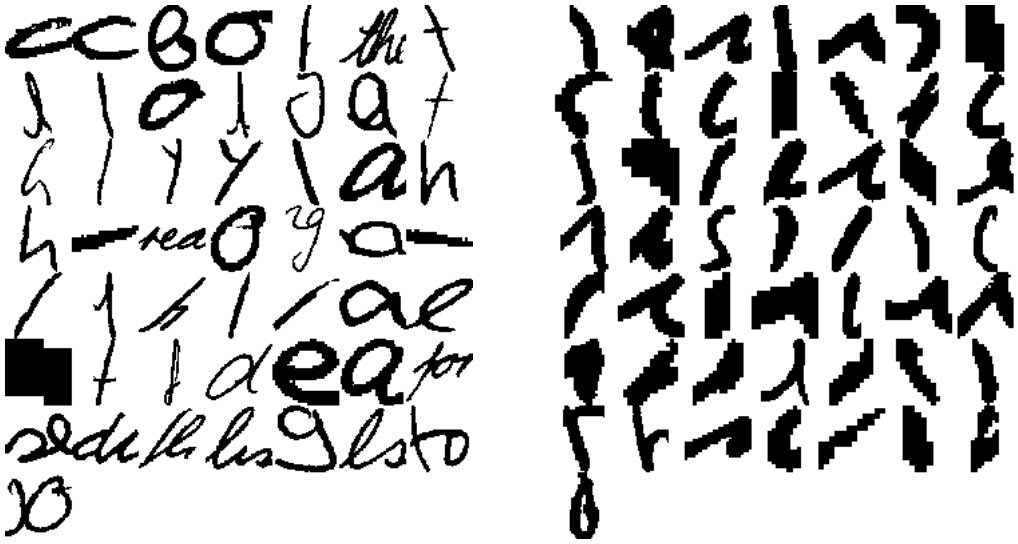
5.4.4 Controls and reference methods

Two controls were also run: the first was a single-stage random selection of g_t graphemes from the dataset pool $\mathbf{G_D}$; the second matched the two-stage approach of the methods under test, randomly selecting g_t graphemes from the baseline codebook of g_s graphemes. These methods should produce equivalent results.

As a further point of reference, a standard clustering method was also implemented. The k-medoids algorithm (Kaufman and Rousseeuw, 1990, p. 40) is very similar to the more common k-means clustering (MacKay, 2003, p. 285): it selects a number of cluster centres, and updates each cluster by assigning each sample to its closest centre, followed by a recentering step. These two stages are repeated until the clusters converge to a stable configuration. The significant difference between k-means and k-medoids is that the latter uses the most central data sample to represent each cluster (the medoid), whereas k-means uses an artificial calculated element (the centroid) which represents the hypothetical sample that would



(A) Scribes dataset sim-min (left) and sim-max (right) selected codebooks



(B) IAM dataset sim-min (left) and sim-max (right) selected codebooks

FIGURE 5.7: Example codebooks generated from similarity-based selection criteria

exist at the actual cluster centre.

In terms of clustering graphemes, this would give codebooks consisting of elements that were not present in the input dataset, rather than the direct grapheme selection that k-medoids provides. This approach also avoids having to define how to compute a composite grapheme (which potentially introduces grayscale), and allows all calculations to be based purely on the grapheme distance matrix without requiring a coordinate-space mapping. The Partitioning Around Medoids (PAM) variant (Theodoridis and Koutroumbas, 2006, p. 636) of k-medoids was therefore implemented in MATLAB, using the standard image correlation as a distance measure and random selection to initialise cluster centres.

5.4.5 Experiment Methodology

As outlined in Section 5.4.1, the feature extraction-based methods (PCA, LDA, and LDA-sparse) require coefficients from an original set of feature vector data, so these experiments were set up in two stages. The baseline experiments (see Section 3.5.1 for details) were used as initial 'source' experiments, and the necessary feature data was drawn from these results. Each selection method is run on the source experiment codebooks of size g_s to produce target codebooks of size g_t for testing, where $g_t < g_s$.

As prior experiments have shown that 200 grapheme features perform well, the experiment was run at two scales with codebooks of size 200 as both the source and target codebook sizes, i.e. target codebooks where $g_t = 200$ were selected from a source codebook where $g_s = 500$ (large-scale), and for the small-scale experiments $g_s = 200$ and $g_t = 50$.

For each dataset, eleven selection methods were tested and eight runs were made of each size experiment on each dataset, giving 32×11 target codebook experiments (in addition to the 32 baseline experiments) for a total 352 experiments run. As before, grapheme comparison was implemented using simple pixel-wise image correlation to generate the feature vectors and the Euclidean-

distance nearest-neighbour algorithm is used for classification. Leave-one-out cross-validation is used for all experiments.

In summary, the selection methods tested are as follows:

- PCA-based
- LDA-based
- minimise similarity
- maximise similarity
- maximise range of similarities
- maximise spread of similarities
- select an even range of similarities
- similarity-spread/LDA hybrid
- Reference: k-medoids clustering
- Random control: matched sub-selection
- Random control: direct target-size selection (baseline)

The results plotted in Figures 5.8 and 5.9 show the identification accuracy of each method, ordered by mean grapheme similarity, and Figures 5.10a – 5.11b show the grapheme similarity distributions in more detail. Unless otherwise noted, the figures given are the mean of the 8 runs, $\pm$ 1 standard error.

5.4.6 Results and Analysis

The results of these experiments fall into two categories which will be discussed in turn. The first is the identification accuracies of the target codebooks, relative to each other and to their source codebook baseline results. The second is the

distribution of grapheme similarities found within each selected codebook, and their interaction with identification accuracy.

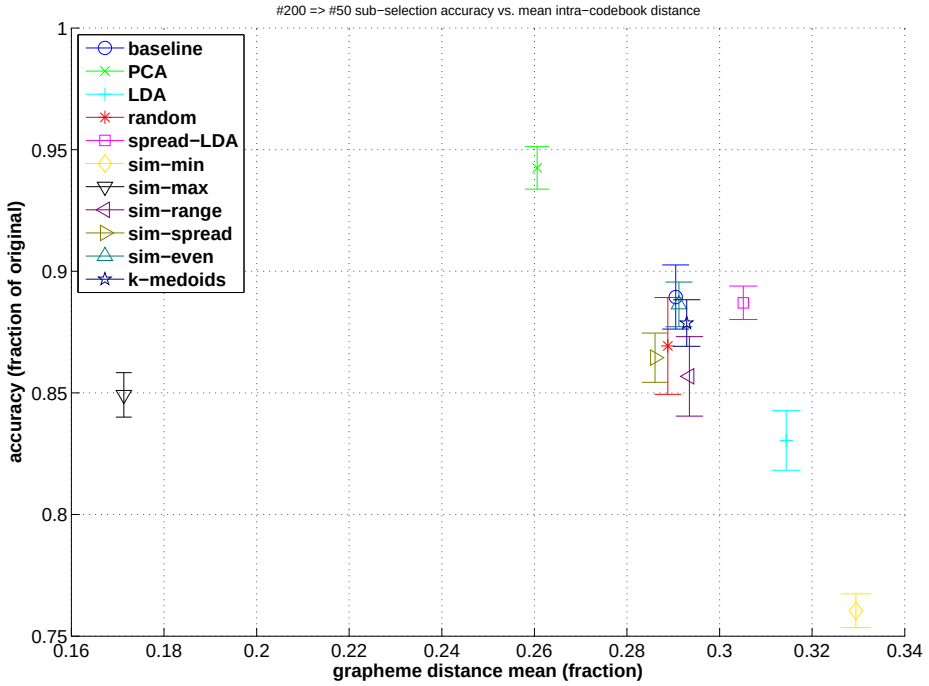
Identification accuracy

Figures 5.8 and 5.9 show the mean and standard error of all 8 runs as a fraction of the baseline identification accuracy of the source codebook. Relative results were used here to exclude the variation due to differences in baseline performance, although a later comparison with absolute accuracies shows that source and target codebook performance is uncorrelated. Overall, the pattern of selection method performance is broadly similar across datasets and experiment scales.

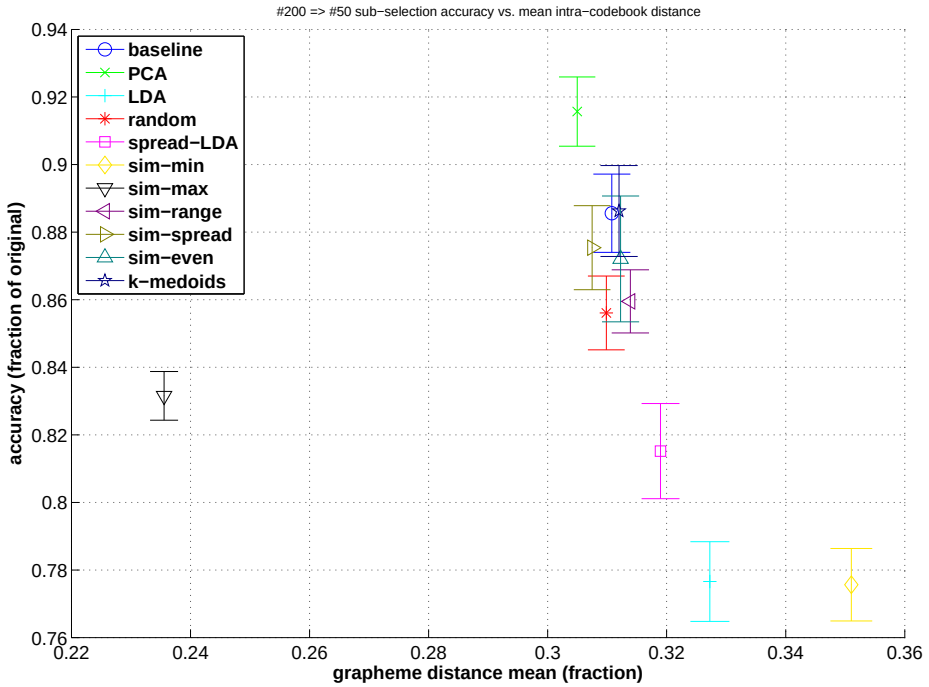
Relative to the source codebooks, identification performance is rarely increased over the source baseline results. Increases occur only in the large-scale IAM-dataset experiments, although source performance is approximately matched by PCA-based selection in the large-scale medieval-dataset experiments as well. No selection method on the small-scale experiments meets this level, with typical performance in the range 80–90% of source codebook accuracy across both datasets.

Comparing selection methods against each other, there are again many similarities across all four experiments. Extremes of similarity range (sim-min and sim-max) and LDA-based selection perform poorly across all experiments. Sim-min is the only selection method which performs significantly worse than random selection in all experiments, while sim-max significantly underperforms random on both large-scale experiments. This supports the hypothesis that a range of grapheme similarities is required for high codebook performance. Sim-max typically performs better than sim-min, i.e. of the two extremes, a codebook composed of very similar graphemes has greater discriminatory power than a codebook of very varied 'outlier' shapes. However, this distinction is not significant on the large-scale medieval-dataset experiment.

PCA-based selection is the best-performing method overall, with behaviour

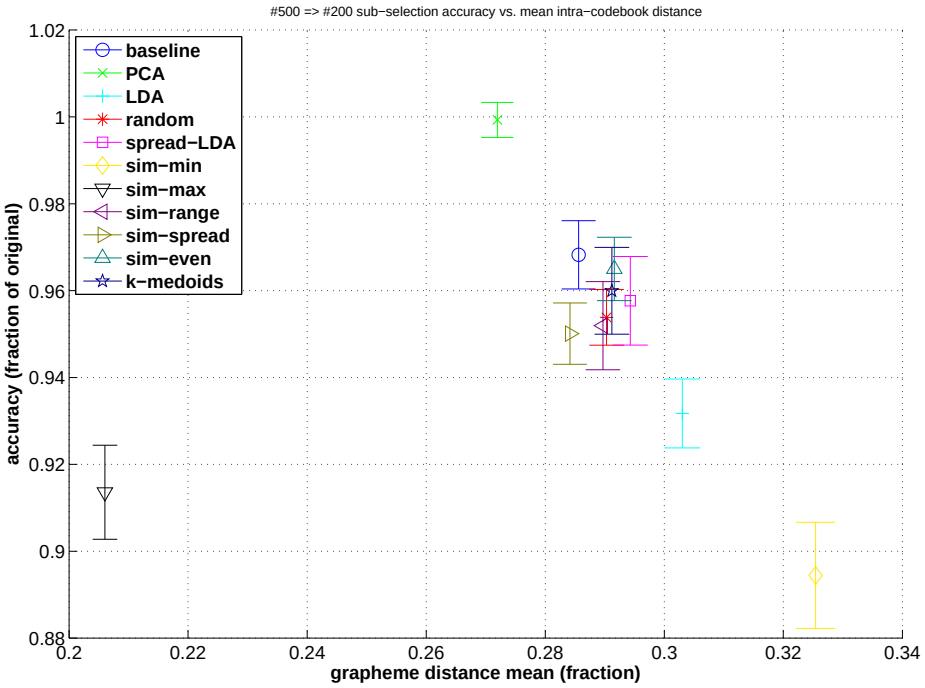


(A) Performance on the small scribes experiment

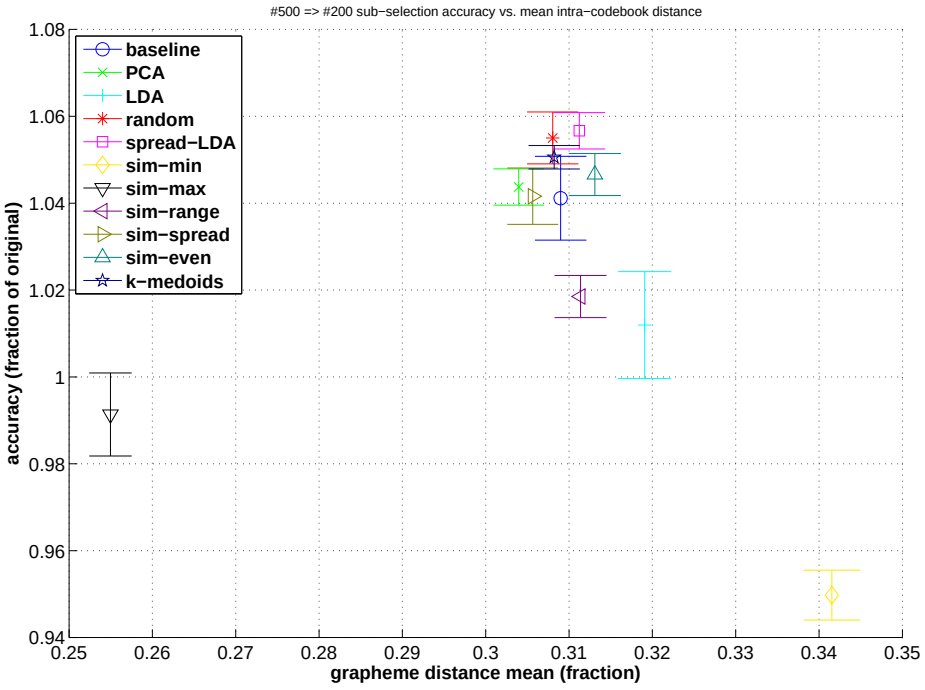


(B) Performance on the small IAM experiment

FIGURE 5.8: Classification accuracy relative to source codebook baseline accuracy, mean of 8 runs (± 1 standard error)



(A) Performance on the large scribes experiment



(B) Performance on the large IAM experiment

FIGURE 5.9: Classification accuracy relative to source codebook baseline accuracy, mean of 8 runs (± 1 standard error)

that varies between the datasets. On the medieval dataset, PCA performs significantly better than all other selection types by approximately 2.5 and 4 percentage points on the large- and small-scale experiments respectively. On the small-scale IAM experiment PCA selection has a weakly significant lead of less than 2 percentage points, and on the large-scale experiment it is one of six selection methods with indistinguishable identification performance.

The remaining selection methods typically cluster closely in accuracy range, with sim-max at the lower end of this group. There is no consistent significant difference in identification accuracy between the random controls and any of the clustering or similarity-balancing implementations on any of the experiments. Isolated deviations from this pattern are atypically low performances of LDA-spread and sim-range on the small- and large-scale IAM experiments respectively, and a difference in means between the two random selection methods on the small-scale IAM experiment which exceeds the 1 standard-error bracket. However, all these results fall well within their respective 95% confidence intervals and are unlikely to be significant in any way.

Of final note is that LDA-spread selection performs better than purely LDA-based selection on all experiments, highly significantly so on the small-scale medieval and large-scale IAM experiments. This supports the hypothesis that increasing the grapheme similarity range of a selection method improves its performance.

Grapheme similarity distributions

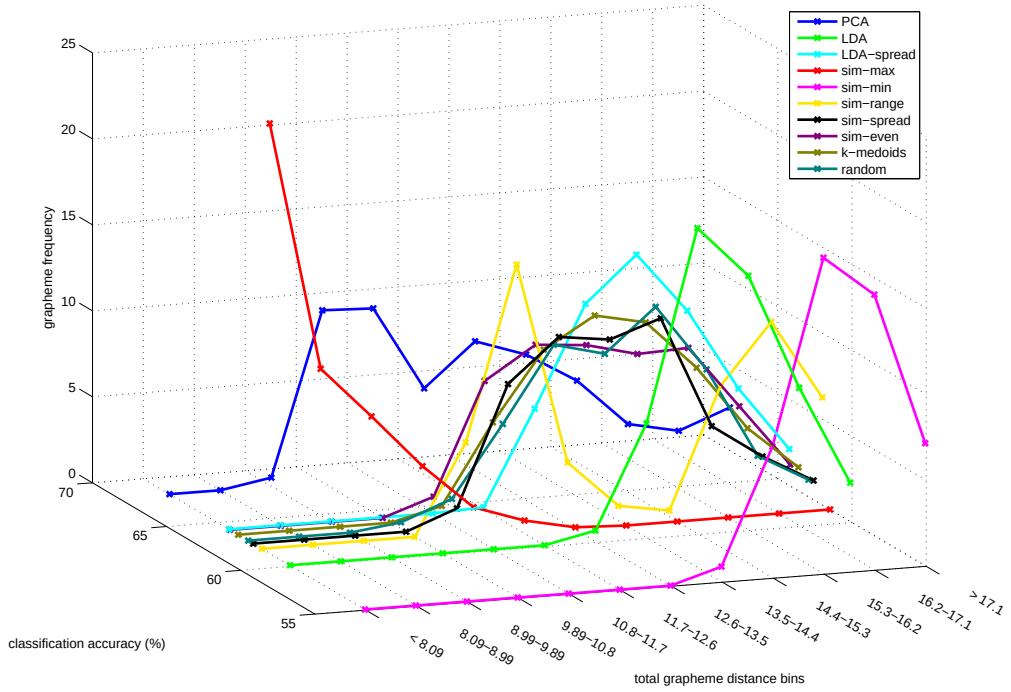
In Figures 5.10 and 5.11, the grapheme distance distributions for each target codebook are averaged over the 8 runs and plotted against the mean identification accuracy. These histograms were generated using a similar method to those in Section 5.4.2: the column-sum of the grapheme image distance matrix gives, for each grapheme, the total distance to all other graphemes in its codebook; typical

values are obviously higher for larger codebooks. To generate the histograms shown for each experiment, the grapheme distances for all target codebooks were pooled and sorted, and the top and bottom 5% were placed into the two end bins. The remaining central range is divided into 10 equally-spaced bins.

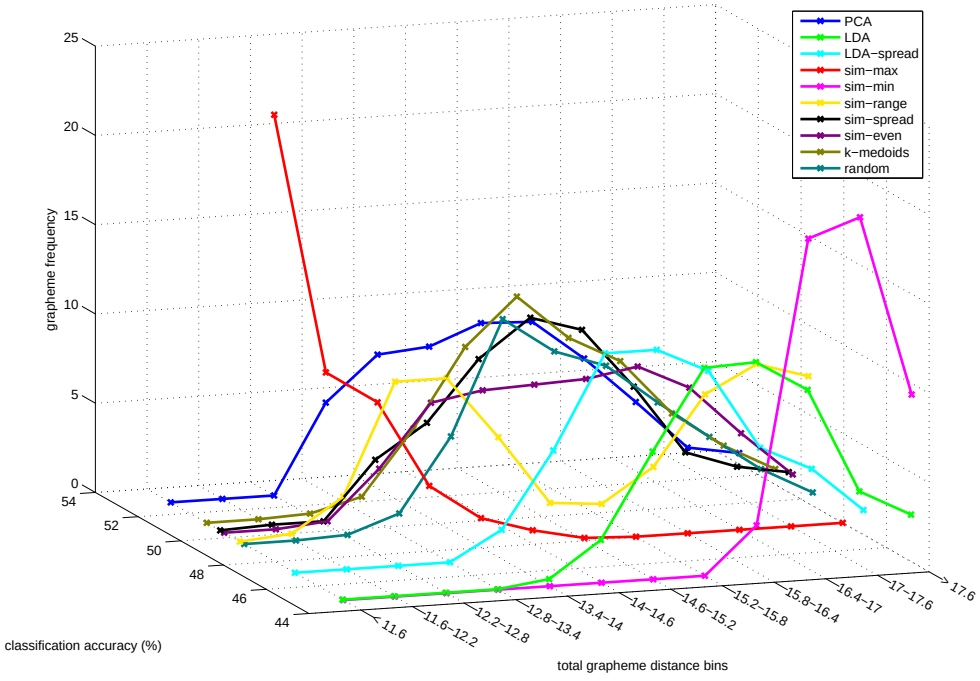
The first distinction between the IAM and scribal datasets is the range of grapheme similarities spanned. The IAM dataset generates codebooks whose graphemes are typically less similar to each other (at both scales) than the scribes dataset. The upper ranges of the similarity distributions are close across both datasets, but the IAM dataset has substantially higher minima, giving a range that covers roughly the top 65% of that of the medieval dataset for both large-scale and small-scale experiments. This supports the hypothesis that medieval writing is overall less varied than modern handwriting.

The majority of selection methods, especially those with similar performance to random-selection, have distributions of a slightly flattened bell-curve shape. Sim-min, sim-max and sim-range produce the most differentiated distributions. The two min- and max-based peaks of the sim-range method are clear on all datasets, as is the 'flattening' effect of the sim-even method. However, none of the similarity-balancing methods have a significant representation of graphemes with low total distance. Additionally, on the scribal dataset at both scales PCA is substantially better at reaching this end of the range than any similarity-range implementation. On the IAM dataset there is no clear difference between PCA's similarity distribution and that of the majority of selection methods, although it could be argued that PCA on the small-scale IAM has somewhat better representation of low-distance graphemes.

As grapheme selections are based on the distribution of the source codebook, it is hard to predict the eventual distance distribution within the target codebook, i.e. the indirection means target similarity distributions are hard to control. The similar/low-distance end of the range is particularly hard to reach, as including a few outlier graphemes disproportionately increases the total-distance measure



(A) Performance and similarity distributions for the small scribes experiment



(B) Performance and similarity distributions for the small IAM experiment

FIGURE 5.10: Classification accuracies w.r.t. codebook similarity distributions, mean of 8 runs ($\pm$ 1 standard error)

for all graphemes in the target codebook. This effect explains why the similarity-range implementations are clipped at the lower end. It is most clearly seen in the distance between the sim-min distribution and the lower peak of the sim-range distribution: despite an overlap in codebook content of approximately half, the lower sim-range peak has been shifted substantially away from the low-distance end of the range.

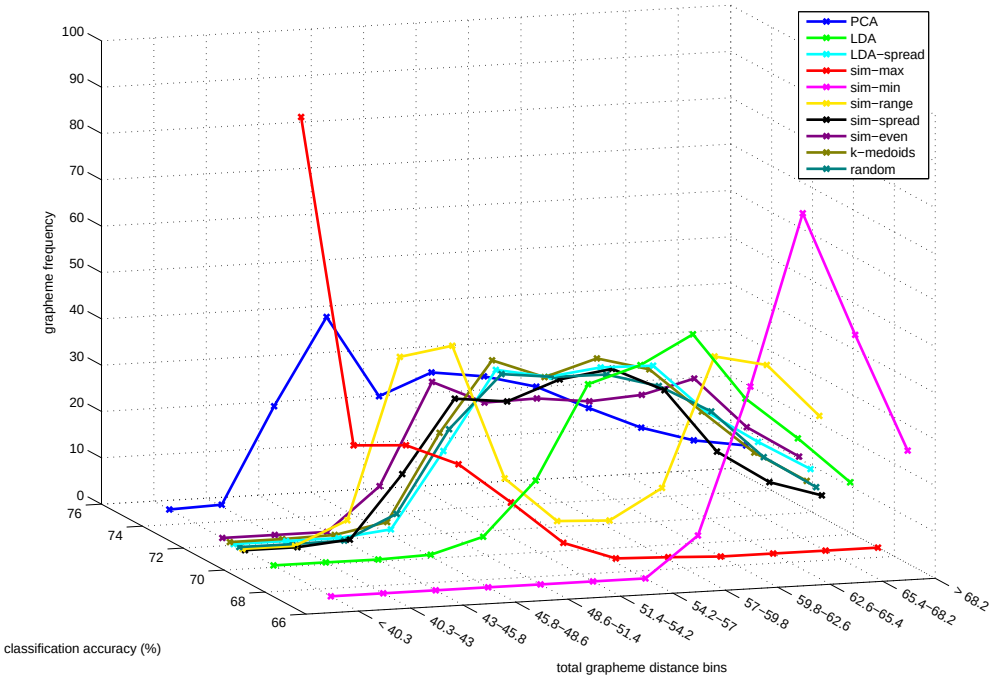
Looking at the hybrid method tested, LDA and LDA-spread have fairly similar distributions. LDA itself seems to skew the grapheme choice substantially towards the dissimilar. This effect is the exact opposite of the hypothesis proposed in Section 5.4.3 , meaning the attempt at 'spreading' the graphemes has had little effect at decreasing similarities, and rather a small effect of introducing a few more-similar graphemes. Despite this, LDA-spread selection generally performs better than LDA, likely due to sim-min being the worst-performing strategy possible.

Overall, these distributions support the hypothesis that the grapheme-similarity distribution of a codebook is a factor in its identification performance, and that a range of grapheme similarities is necessary.

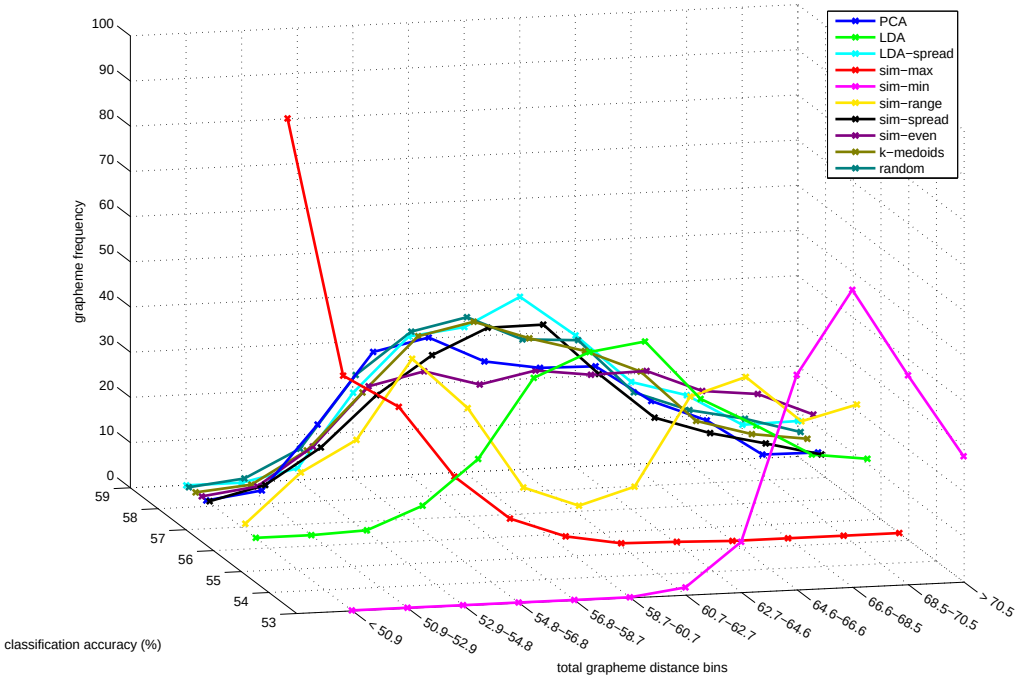
5.5 Conclusions

In this chapter, three feature extraction methods have been applied to the grapheme codebook. ICA and LDA perform poorly, with analysis suggesting that sparse feature vectors may be a contributing factor. PCA was found to perform significantly better, matching the raw classification accuracy on both datasets using approximately 33% of the number of dimensions on the medieval dataset, and approximately 60% on the IAM dataset.

However, feature extraction obscures the intuitive link between grapheme codebook features and the writing fragment samples they originate from. Although the extracted dimensions can be visualised by weighting grayscale grapheme images,



(A) Performance and similarity distributions for the large scribes experiment



(B) Performance and similarity distributions for the large IAM experiment

FIGURE 5.11: Classification accuracies w.r.t. codebook similarity distributions, mean of 8 runs ($\pm$ 1 standard error)

these are too indistinct to be used directly as codebook components. A range of methods for grapheme selection were therefore tested. Some of these were based on the feature extraction methods to find the individual graphemes that contribute most to the top extracted features, and some were based on codebook-level criteria based on grapheme similarity distributions. PCA-based selection again performs best, and by a significant margin on the scribal dataset. Most selection methods, including clustering, have similarity distributions and identification performance close to that of random selection. Selection methods focusing on the extremes of the similarity distribution had the worst performance, supporting the hypothesis that a range of grapheme similarities is required for high performance.

Chapter 6

Classification

Classification is the final aspect of the identification process. The work in this chapter considers two variations on the implementation of simple nearest-neighbour classification to this point. Section 6.1 examines the effect that manuscript or document origin has on identification accuracy. Given the formal, stylised nature of most manuscripts in the period under consideration, this work tests the hypothesis that the style of the physical manuscript which a sample originates from is a stronger confounding factor in medieval than in modern writer identification. Quantifying this effect can help avoid overestimation of identification accuracy in same-document training sets. Section 6.2 uses the information discarded during nearest-neighbour classification to add a verification layer to the identification system. This allows estimation of the reliability of each writer label prediction originally made.

6.1 Classification strategy

The following experiment focuses on a document-specific aspect of classification: the effect that the manuscript or document style has on the overall writing style, and thus on writer identification accuracy.

This section considers the classification strategy used in writer identification.

The standard method of cross-validation is typically used to ensure reliability of results whilst maximising use of the training dataset. It divides the training set into n 'folds', each of which are tested separately against the remaining data. The strategy known as 'leave-one-out' has been used throughout the experimental work in this thesis. It is a special case of n-fold cross-validation where the number of folds equals the number of training samples, resulting in a single test sample being compared against all remaining training data in each round.

Whilst this is general good practice, it does not take into account a relevant domain-specific issue. In historical documents, the style of the writing is a combination of the scribe's personal style, and the professional styling of the document being composed. The physical manuscript copy that a historical image sample originates from is therefore an important factor in determining writing style and classification accuracy. Grouping the data by document forces the classifier to generalise over all the writing styles a scribe has produced, without taking advantage of document-based rather than writer-based similarities. The writing in the modern IAM dataset contains samples of different content types or genres: news articles, letters, novels, etc., but all samples are copied by writers in their natural personal style. It is therefore expected that there will be little impact from factoring in the imposed document writing style on this dataset.

The aim of this experiment is therefore to correct for the effect of manuscript style in historical writer identification accuracy, to produce a more realistic prediction of how well the trained system may perform on unseen data. It will estimate the size of the document-style effect by modifying the classification strategy from the standard leave-one-out. Instead of testing one sample and using all other data to train, this strategy excludes from the training set any samples by the same writer from the same physical manuscript or document as the sample under test. This will produce a pessimistic estimate of the system's identification accuracy as it requires the classifier to work without same-document style information, and with a reduced amount of total training set data. There may also be same-document

distractors: samples from the same manuscript written by a different scribe, but with a matching document style.

As the historical dataset has varying numbers of image samples per scribe and per manuscript, these issues unfortunately affect different scribes to different degrees; however this is a realistic test situation which will provide valuable domain-specific information. These experiments will also be run on the IAM dataset to compare the magnitude of the effect on identification accuracy, and with a control experiment to estimate the effect of reducing the training set.

6.1.1 Methodology

In the scribes dataset, samples consist of page or part-page images, grouped by document: here, this is the physical manuscript or book from which they were drawn. For the experiments in this Section, only a subset of the writers are used: those who meet the criterion of having image samples from at least two documents. This subset contains 24 writers, with a total of 328 images split into 101 documents. There are 1–19 samples per document (median 2), and 2–14 documents per writer (median 3).

The IAM dataset consists of single-line image samples, segmented from a page of freehand text. As these pages and their content are independent, each page is considered to be its own document with the line images as samples. There are 93 writers, 465 documents, and 4261 images in total, with 3–12 samples per document (median 9), and 5 documents per writer. As all writers have more than two documents, the entire dataset is used. On the IAM dataset therefore, a line image tested will need to be matched against a line from another page to be classified correctly; on the scribes dataset, a page sample will need to be matched against a page from a different manuscript.

The experiment will be run eight times using the standard methodology summarised in Section 3.5.1, with the following exception: three classification modes will be tested. The first is the standard leave-one-out (LOO). The second is the

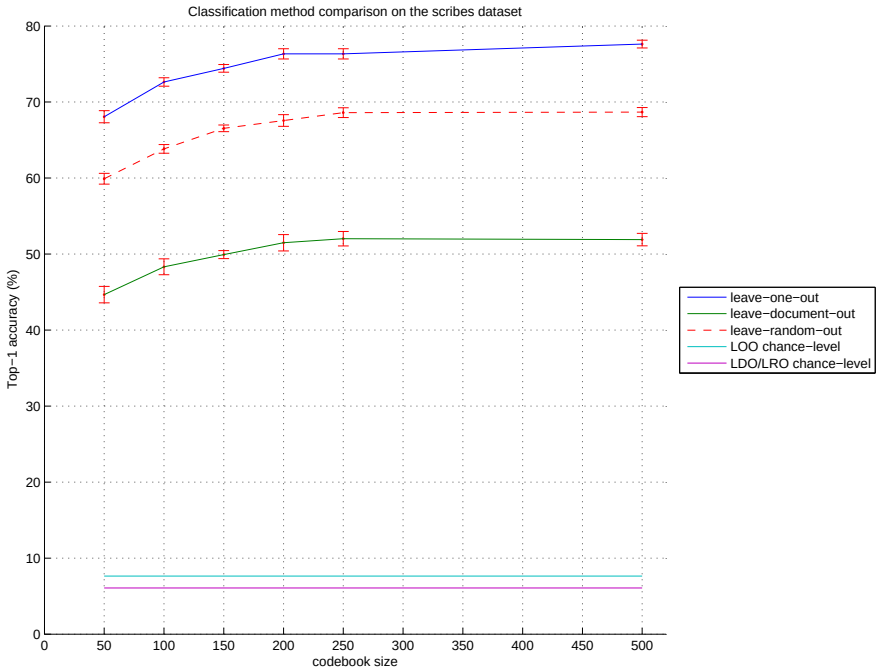
proposed leave-document-out (LDO) method: all samples by the same writer and from the same physical document as the test sample will be excluded from the training set for that fold. Apart from the style issues being analysed, this also makes the identification problem statistically harder as the proportion of 'correct' (i.e. same-writer) matches in the training set is decreased.

To quantify this effect, a third classification method is also tested: the leave-random-out (LRO) strategy is a control that accounts for these changes in training-set size. This method counts the n samples with the same document as the test image, but instead of excluding them all as in LDO, it randomly chooses n samples from that writer to exclude. This produces a training set that has identical correct-writer:incorrect-writer sample ratios as LDO (and therefore the same level of identification by chance), but without the explicit exclusion of all same-document samples. This may of course happen incidentally, and is more likely to occur in the scribes dataset where most writers have fewer attributed samples than the IAM dataset.

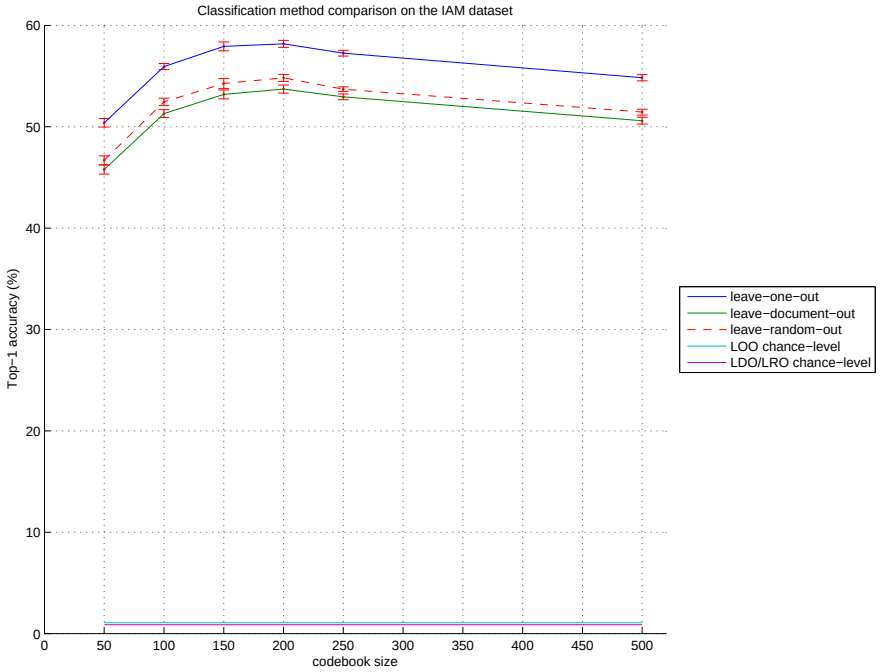
The leave-one-out mode gives training sets of 327 and 4260 image samples on the scribes and IAM datasets respectively. In leave-document-out or leave-random-out mode, the training set for each fold will contain 309–327 and 4249–4258 samples for the scribes and IAM datasets respectively, depending on the number of samples attributed to the same document as the test sample. In the presented results, the per-sample chance-level calculations described in Section 3.4.3 have been adapted to match the new methodology, and both chance-levels are indicated separately in the results.

6.1.2 Results and Analysis

Both datasets show a drop in actual and chance-level identification accuracies using LDO and LRO compared to the standard leave-one-out method. The small chance-level drops reflect the reduced probability of randomly choosing a same-writer sample from the training set as under both LDO and LRO the number of



(A) Classification strategy comparison on the medieval dataset



(B) Classification strategy comparison on the IAM dataset

FIGURE 6.1: Classification strategies on each dataset, mean of 8 runs $\pm$ 1 standard error

these samples is reduced, while the number of different-writer samples remains the same. It is also clear that for both datasets, codebook size is not a significant factor: the percentage-point drop in identification accuracy is consistent across the range.

In the interest of completeness an additional control method was tested, although for clarity the results are not plotted here. This method also selected n samples to remove (where n is again the number of same-document/same-writer samples), but these were randomly taken from the whole training set rather than being restricted to a single writer. Mean identification accuracies and standard errors for this method were essentially identical to LOO across both datasets and all codebook sizes. This shows that simply reducing the total training set size confers neither a positive nor a negative effect, which confirms that the drop in identification accuracy using LRO is due to the difference in the same-writer:different-writer ratio, rather than the overall training-data reduction.

On the scribes dataset, the leave-document-out method has a much lower identification accuracy than leave-one-out, with a drop of approximately 24 percentage points (Figure 6.1a). The results from the leave-random-out control experiment show that only approximately 8 percentage points of this can be attributed purely to a reduction in same-writer training data. This strongly suggests that the remainder of the decrease is due to the inability to distinguish the writer's style across different documents.

On the IAM dataset, the drop between the LOO and LDO methods is much smaller, at approximately 4.8 percentage points (Figure 6.1b). It is also clear that the majority of this drop is accounted for by the LRO method, i.e. the drop is almost completely due to the reduction in the amount of same-writer training data. The effect of the physical document style appears to be minimal, including any effect the type of content may have had on the writer's style.

6.1.3 Conclusions

These experiments have shown that the personal writing style present in the historical data is substantially obscured by the style of the document a writing sample originates from. The physical manuscript is therefore a highly significant variable which must be taken into account in the analysis of medieval scribal texts.

In contrast, the document style had very little effect on writer identification in modern handwriting. This suggests that even for small samples, variation in the content-type of writing samples is likely to have minimal effect on the measurable writing style.

The following section will move from classification method considerations to the next stage in the process – given a set of identified samples, is it possible to determine the proportion of a dataset which can be classified confidently?

6.2 Verification thresholds

The classifier used in all these experiments is the nearest-neighbour classifier, which finds the single closest match in the training set and attributes the test image to its writer. The identification accuracy reported from each of these experiments is simply the proportion of these assignments that was found to be correct, with no additional information regarding how 'good' each match was.

As this information is useful for deeper analysis, the following section proposes using the distance between the test image and its closest match to determine the quality of the match, and how confident the system is in its identification. Aggregating these distances over the whole data set can also provide information on the proportion of data that can be confidently identified.

6.2.1 Problem Background

Although writer verification is a substantial field in its own right (most notably in signature verification, see Section 2.1 for a summary), current work in writer identification rarely offers dataset or system performance indicators beyond simple identification accuracy. A notable exception to this is the work of Bunke et al. (Schlapbach and Bunke, 2007b, 2006a, 2004b), where a rejection mechanism for a Hidden Markov Model-based writer identification system has been implemented. Four confidence measures have been tested, which are used to reject image samples that do not meet a given confidence threshold. By rejecting the most ambiguous 21% of the input images, the remaining data could be classified with 99.9% accuracy (Schlapbach and Bunke, 2007b).

Similar measures of the system's level of confidence in its identity attributions are particularly useful to historians in the further analysis of manuscripts, at both the level of individual images and the overall dataset.

The following work describes an analysis of the baseline results (Section 3.6) on the medieval and IAM datasets. This method is also more broadly applicable to nearest-neighbour results generated from any set of features.

6.2.2 Method

This method builds an additional layer onto the existing stage of identification at which unknown samples are labelled with the writer they match most closely. The secondary stage augments this with an additional piece of information: how confident the system is that this writer is correct. The higher the value, the more likely the writer label is to be correct, with very low values suggesting that the label is incorrect, and intermediate values indicating insufficient evidence to decide either way.

By applying a threshold to these values, a prediction can be made as to whether the assigned writer is correct or incorrect. The combination of primary classifi-

		Identification	
		Correct	Incorrect
Verification	Correct	True Positive	False Positive
	Incorrect	False Negative	True Negative

TABLE 6.1: Accuracy classes for primary classification and secondary verification

cation labels (i.e. writers) and secondary predictions about them will fall into the following categories:

- Writer label is correct, and predicted to be correct (accurate prediction)
- Writer label is correct, but predicted to be incorrect (inaccurate prediction)
- Writer label is correct, but prediction is unsure (no prediction made)
- Writer label is incorrect, but predicted to be correct (inaccurate prediction)
- Writer label is incorrect, and predicted to be incorrect (accurate prediction)
- Writer label is incorrect, but prediction is unsure (no prediction made)

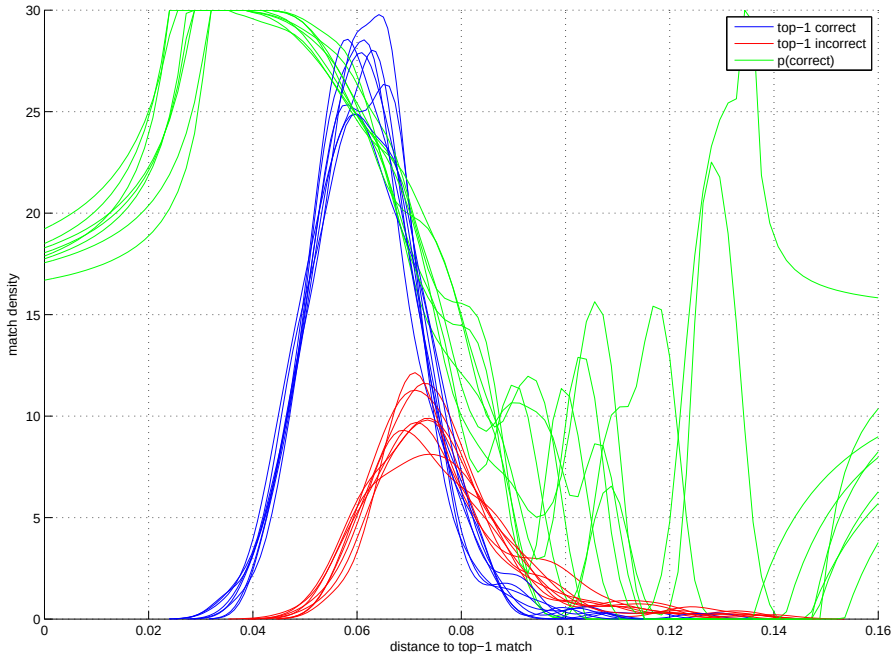
The more confident the system must be before declaring a match, the more likely the match is to be correct, but a correspondingly smaller proportion of the dataset will meet that threshold. The inverse of the same confidence threshold can also be used to actively highlight the samples that the system has identified incorrectly. For example, setting the threshold at 70% means that samples with a confidence level of 70% or more will be predicted to have correct writer labels, and samples with a confidence of 30% or less will be predicted to have incorrect labels. The remaining samples have confidence c where $30\% < c < 70\%$, and will be rejected as ambiguous.

The retained data therefore falls into the four results categories for which an accurate/inaccurate prediction is made. In verification terms, these categories are respectively defined true positives, false negatives, false positives, and true negatives, as summarised in Table 6.1. In these terms, the primary classification accuracy is the sum of the true positives and the false negatives at a threshold of 50%, i.e. the samples whose labels were assigned correctly, regardless of whether this was accurately verified, with no samples rejected as ambiguous.

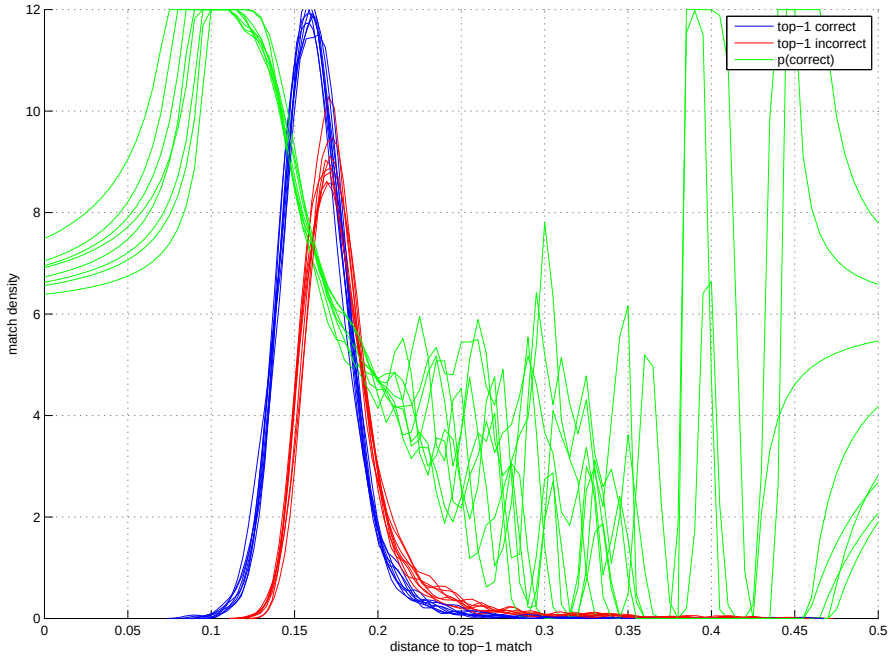
This method considers the distance between a test sample and its nearest-neighbour as an indication of how likely they are to belong to the same writer class. In the identification system, this distance information is discarded. The verification system instead collects these distances from the training data and divides them into two groups according to whether the test sample was correctly or incorrectly classified. The distribution of distances in each group provides information on how frequently particular test–neighbour sample distance ranges result in correct attributions. Given a sample of unknown authorship, the distance to its nearest-neighbour can be used to estimate how likely the nearest-neighbour identification is to be correct. The following section describes the implementation of this process.

6.2.3 Implementation

Given a classified dataset, the verification stage initially estimates the distance-to-nearest-neighbour distributions of the image samples which were classified correctly and incorrectly (Figure 6.2, blue and red plots respectively). This was implemented using the MATLAB ksdensity() function to provide a smoothed distribution. The figures show that the majority of samples are in the range 0.05–0.09 from their closest match in the medieval dataset, and 0.12–0.23 in the IAM dataset. It is also clear that the samples finally identified correctly are typically closer to their match than those incorrectly identified, and that the spread of the IAM dataset is substantially greater than for the medieval data.



(A) Correct/Incorrect distance distributions for the medieval dataset



(B) Correct/Incorrect distance distributions for the IAM dataset

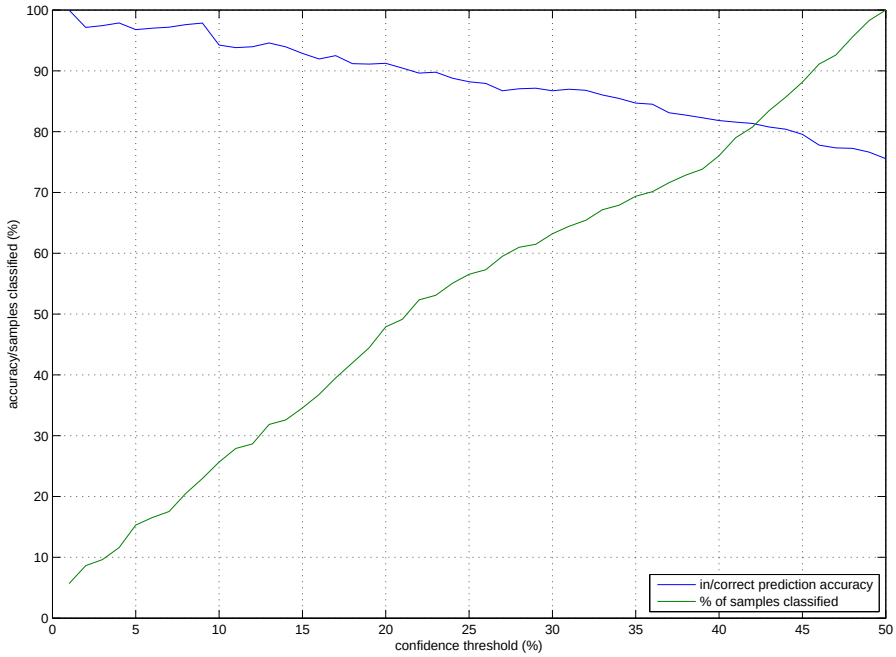
FIGURE 6.2: Correct/Incorrect nearest-neighbour distance distributions for each dataset with prediction confidence, eight runs at codebook size 150

The relative areas under each curve reflect the primary classification accuracy: on the IAM dataset, the mean over the eight runs is 57.9%, and 71.7% for the medieval set. On the IAM data, the area covered by the blue curve is therefore typically slightly larger than the red. On the medieval scribes data, the correct/blue curves account for approximately $\frac{2}{3}$ of the data and the incorrect/red curves account for approximately $\frac{1}{3}$, i.e. covering approximately half the area of the correct curves.

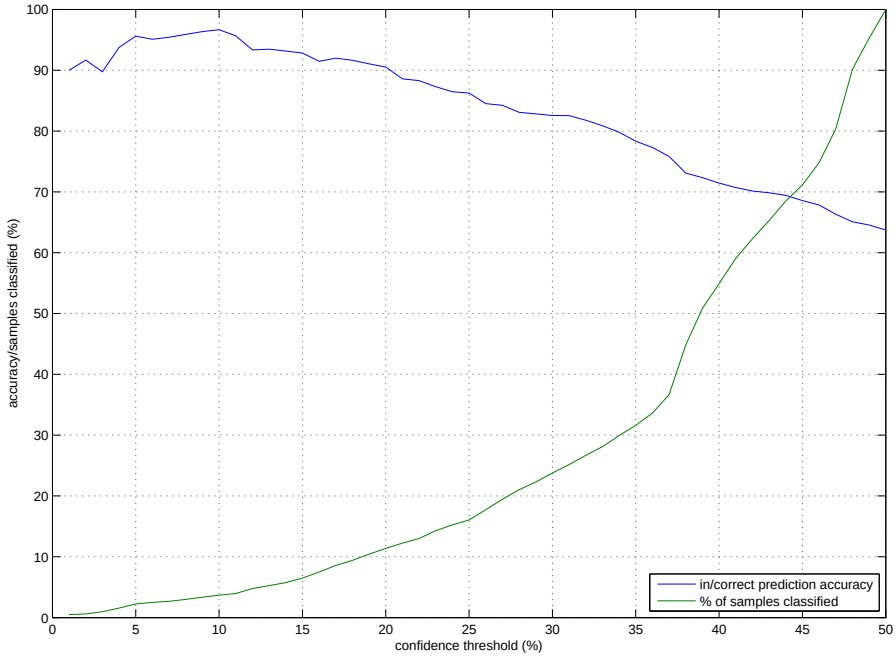
By taking the ratio between the correct and incorrect distributions, the probability that any given nearest-neighbour distance represents a correct or incorrect match can be calculated. This is implemented on a leave-one-out basis, so that the distributions used in the calculation never contain the test sample's distance information. There are three edge-cases to handle where the ratio method is insufficient: the test sample's nearest-neighbour distance may be within the training data range of the 'correct' cases but outside the range of the 'incorrect' cases and vice-versa, or it may be outside the range of both distributions.

In the former case, the confidence assigned to the test sample's label is to be almost completely sure that the primary labelling is correct or incorrect respectively, i.e. one or zero, $\pm$ an uncertainty factor delta. In the latter case, the confidence level is calculated proportional to the distance to the closest endpoints of each distribution. The MATLAB code used is given in Listing 4 in Appendix C.

The green plots in Figure 6.2 show the probability of label correctness indicated by this calculation at each potential distance between a sample and its nearest-neighbour. Values outside the distance ranges which the training data covers are included as examples of the edge-case calculations described. It is clear that 'correct' primary classifications are easier to identify at the lower end of the distance range, and that uncertainty is much higher for larger distances.



(A) Confidence thresholds on the medieval dataset



(B) Confidence thresholds on the IAM dataset

FIGURE 6.3: Confidence thresholds on the medieval and IAM datasets, single run at codebook size 150

6.2.4 Results and Analysis

This Section will discuss typical results obtained from this method, illustrated by a comparative case study of the IAM and medieval dataset baseline results (Section 3.6) for a codebook size of 150. Full results figures are available in Appendix D.

Figure 6.3 shows the trade-off between the proportion of the labels the secondary system can predict accurately (i.e. true positives and true negatives) and the proportion of the dataset that remains after rejecting samples that do not meet the confidence threshold. The vertical scale shows both the proportion of unambiguous samples remaining for label prediction and the secondary prediction accuracy on that subset. Note that no values are plotted for a threshold confidence of zero as no predictions are made with total confidence, leaving zero samples to be classified at this point.

The span of the confidence threshold, or p-value, is limited to half the available $[0, 1]$ range, as its inverse is taken as the threshold for predicting incorrect labels. A confidence level near 0.5 indicates the system is very weakly confident about its verification prediction; thresholds close to zero (or equivalently, one) indicate high certainty.

As a consequence of a two-layer system, two main types of system accuracy are reported. *Identification accuracy* is the initial output: a prediction of which writer label or class to assign to an unknown sample. This prediction can take as many values as there are writers, and the label can be correct or incorrect. *Verification accuracy* is the secondary output: a prediction of whether a sample's identification is correct, incorrect, or ambiguous according to some threshold parameter. Ambiguous samples are *rejected* from the verification system, those that remain are the *attempted* samples that the verification system considers. A sample may be verified correctly when the initial identification is wrong (True Negative), and vice versa (False Positive). Table 6.1 summarises the possible final outcomes for attempted samples. As the accuracy trends over the range of codebook sizes

are very similar a codebook size of 150 has been chosen as a case study, although any size-related variations of interest will be noted. Figures 6.4a and 6.4b illustrate the main accuracy rates of interest on each dataset for 8 runs of this codebook size.

In these Figures, the horizontal **primary accuracy** line indicates the identification rate when all samples are included, i.e. the baseline classification rate reported in earlier Chapters. It represents the performance of the existing one-level identification system.

The solid pink **tp+fn** line is the identification accuracy at each threshold when only the attempted samples are considered.

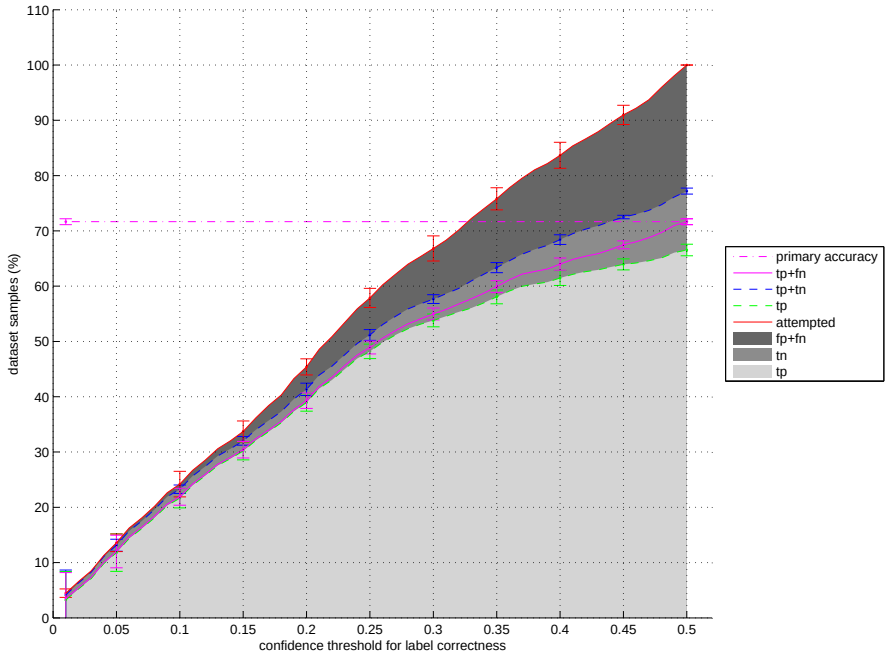
The blue dashed **tp+tn** line is the verification accuracy, i.e. the success rate of the secondary prediction of whether a writer label is correct or incorrect. This is equivalent to the blue curves in Figure 6.3, which show the accuracy on the attempted samples only, whereas Figure 6.4 shows this in the context of the whole dataset (i.e. incorporating both the verification accuracy and reject rates). The distance to the **tp+fn** line below indicates the small but significant advantage in prediction accuracy gained from using this verification system.

The green dashed **tp** line and corresponding light grey area shows only the proportion of the dataset the system was able to predict correctly. This is the most stringent measure of the combined identification and verification system accuracy.

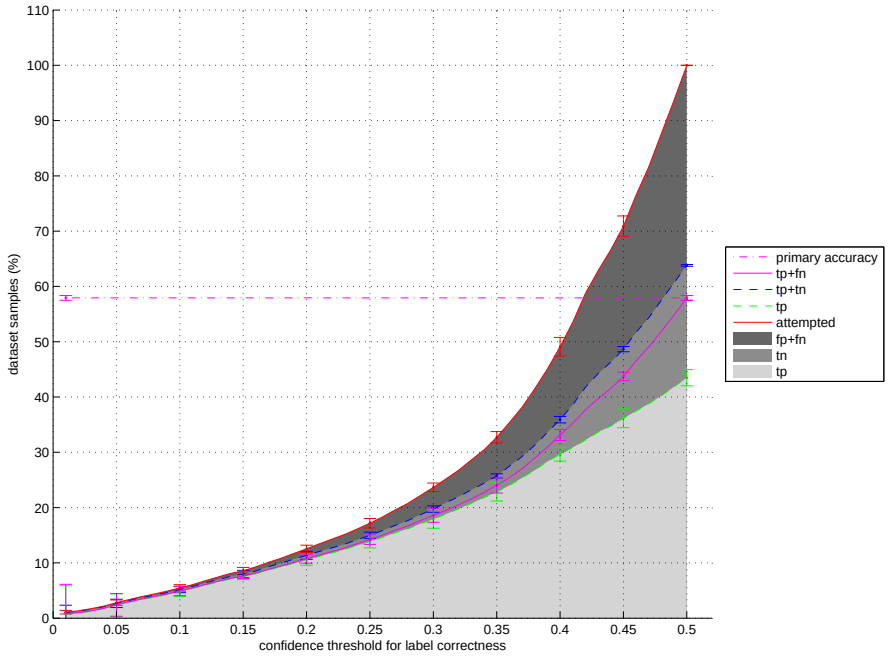
The solid red bounding **attempted** line shows the proportion of the dataset meeting the verification system threshold, as this varies along the x-axis. It is equivalent to the green curves in Figure 6.3. Although Figure 6.3 illustrates a single run only, the small error bars in Figure 6.4 demonstrate this curve shape is typical of the overall result.

The shaded **tn** area shows the proportion of the dataset for which the verification system was able to highlight an identification error made by the identification stage, correctly picking out incorrect writer labels.

The dark shaded **fp+fn** area shows the total verification error rate on the sam-



(A) Error/accuracy rates on the medieval dataset



(B) Error/accuracy rates on the IAM dataset

FIGURE 6.4: Error rates for each confidence threshold on the medieval and IAM datasets as a proportion of the total dataset, 8 runs mean and standard error at codebook size 150

Codebook Size	Accurate Identify & Verify (TP)			Accurate Verification (TP+TN)		
	Rate %	Threshold	Samples %	Rate %	Threshold	Samples %
50	91.33	0.09	15.68	97.85	0.09	15.68
100	87.19	0.09	17.50	98.66	0.02	7.50
150	90.17	0.12	28.52	98.21	0.04	11.30
200	91.47	0.07	19.85	97.29	0.07	19.85
250	93.50	0.07	19.81	97.26	0.07	19.81
500	92.06	0.10	29.01	96.31	0.09	26.94

TABLE 6.2: Scribal dataset peak verification accuracy rates, corresponding thresholds, and samples attempted for two accuracy measures. Mean of 8 runs given for each codebook size.

ples it attempted. It is a combination of the accurate writer label assignments it predicted incorrect, and the inaccurate labels it predicted correct.

It is clear from both these Figures that only a small fraction of either the IAM or scribal datasets can be identified with high confidence. At a threshold of 0.05, only 3% of the IAM dataset and 13% of the scribes dataset samples are attempted by the verification system for a codebook size of 150; if the threshold strictness is relaxed to 0.25, 18% of the IAM dataset and 58% of the scribal dataset samples are attempted. Verification accuracy only exceeds baseline identification accuracy when the threshold stringency is lowered to 0.4–0.45 on the scribal dataset, and 0.45–0.5 on the IAM dataset. However, verification performance always exceeds the initial identification performance on any attempted subset of samples, showing the ability to find incorrect identifications is still valuable. At a threshold of 0.5 where the full dataset is passed to the verification system, identification accuracy is exceeded by 4–8 percentage-points on the scribal dataset and 6–11 percentage points on the IAM dataset, depending on codebook size.

Although its accuracy boost is greater, the IAM dataset presents a harder overall verification problem in two ways: absolute verification accuracies are lower than their scribal dataset equivalents at all thresholds, and the rate of increase in accuracy as the threshold is relaxed is slower, giving a concave curve (as opposed to the medieval dataset's convex shape). The IAM dataset's lower identification

Codebook Size	Accurate Identify & Verify (TP)			Accurate Verification (TP+TN)		
	Rate %	Threshold	Samples %	Rate %	Threshold	Samples %
50	73.44	0.22	7.13	91.00	0.10	2.29
100	87.04	0.11	5.12	95.14	0.03	1.60
150	88.27	0.12	6.76	95.49	0.06	3.37
200	88.28	0.12	7.53	95.61	0.08	4.84
250	88.79	0.12	7.92	94.83	0.07	3.95
500	89.33	0.10	6.09	95.83	0.08	4.72

TABLE 6.3: IAM dataset peak verification accuracy rates, corresponding thresholds, and samples attempted for two accuracy measures. Mean of 8 runs given for each codebook size.

rate is almost certainly a contributing factor to at least the first of these characteristics. The variation in True Positive rates with increasing codebook size also reflects the identification accuracy trend, peaking at a codebook size of 150 on the IAM data before dropping off noticeably (e.g. at a threshold of 0.5), whereas the scribes dataset rises slowly with increasing codebook size.

At the opposite end of the threshold scale, Tables 6.2 and 6.3 show the True Positive accuracies of approximately 90% and verification accuracies of approximately 95% (IAM dataset) or 97% (scribal dataset) are achievable on a sufficiently limited fraction of each dataset. Except for the smallest codebook size of 50 graphemes, similar peak accuracies are noted at all codebook sizes. On the IAM dataset, the corresponding attempted dataset fractions are fairly level. This suggests that beyond an initial codebook size of 50, increasing the number of grapheme features does not significantly improve IAM dataset performance. In contrast, although the peak verification rates on the medieval dataset remain flat, the proportion of the dataset meeting this threshold requirement generally increases with codebook size, particularly for the TP+TN accuracy criterion.

6.2.5 Summary

Introducing a simple verification layer to this identification system resulted in a small but significant increase in the output accuracy of the combined system. Each identification prediction is now augmented with the system's level of confidence, and a single adjustable threshold can be set to decide whether an attribution is trustworthy, untrustworthy, or ambiguous. Relaxed thresholds (near 0.5) maximise the proportion of the total dataset classified correctly, while stringent thresholds allow very high classification accuracies on a small subset of the data. Changing the codebook size gives a similar distribution of system accuracies at each step, and trends in verification accuracy broadly follow the underlying identification accuracy. Possibly due to lower identification performance, increasing the codebook size on the IAM dataset beyond 100 graphemes does not significantly improve either peak verification accuracy or the proportion of the dataset that meets the required threshold. However, the scribal dataset does benefit from increased codebook size as this increases the proportion of data for which the system is highly confident.

6.3 Conclusions

In this Chapter, two separate parts of the classification process have been refined. Section 6.1 demonstrated that the standard leave-one-out classification strategy easily results in significant overestimates of system accuracy on historical data, while making only a negligible difference to the modern IAM handwriting. This is due to identifications that are strongly affected by the document aspects of a sample's writing style, rather than the purely writer-specific. The work presents a method for analysing a dataset which quantifies the impact of this particular confounding factor.

Section 6.2 adds a simple verification layer and identification confidence to each writer label prediction. This gives a significant increase of 4–8 percentage-

points and 6–11 percentage points on the scribal and IAM datasets respectively.

Together, these techniques allow a deeper analysis of the modern and medieval datasets which highlight their similarities and domain-specific differences.

Chapter 7

Conclusions

The overall hypothesis presented in this thesis was that medieval and contemporary writing would respond differently to writer identification techniques.

To test this hypothesis, a series of experiments were executed covering aspects of each stage of the writer identification process, on two datasets of writing images from each of the medieval and contemporary periods. These experiments were designed primarily to discover areas of the identification process, where medieval and contemporary data require different handling, but also to provide a more in-depth study of the data than is typically available in the writer identification literature.

The results of these experiments are summarised in Section 7.1, and indicate that the medieval and IAM datasets do differ in their response to several of the techniques developed and tested in this thesis. The work also demonstrates that some techniques are equivalently applicable to both datasets, and additionally proposes a number of novel methods of writer identification analysis. Section 7.2 identifies limitations of the work presented and potential areas for further work, and Section 7.3 concludes this thesis.

7.1 Contributions

The conclusions drawn in this work are discussed under each of the three categories mentioned earlier, along with the areas open for further work. Sections 7.1.1 and 7.1.2 summarise the results which demonstrate dataset differences and similarities respectively, and Section 7.1.3 addresses the methodological proposals made.

7.1.1 Dataset differences

Differences in the responses of the scribes and IAM datasets were apparent throughout the codebook-based writer identification process.

In the initial stage of segmenting the graphemes from the full ink-trace image, a combination of ligature-focused and character-body-focused graphemes improved IAM dataset accuracy by 12% over the standard character-body only segmentation, whilst having no significant improvement on the scribes dataset (Section 4.3). The ligature-focused graphemes segmented from the scribes dataset therefore contain no writer-specific information (in net) that is not already present in the character-body data. It is however likely that the advantage gained by the IAM dataset from this 'bootstrapping' may be at least partially influenced by the significantly smaller number of graphemes per sample when splitting by a single criterion only.

In normalising graphemes, aspect ratio was found to be a writer-specific characteristic of the IAM dataset, but not of the scribal dataset (Section 4.2). This is likely due to character aspect ratio being largely determined by font in medieval writing, rather than by personal preference as in modern freehand. Removing text aspect ratio in medieval data is therefore likely to remove an element of document-style interference, improving scribe identification accuracy.

When comparing grapheme bitmap images, the translation-invariant cross-correlation distance was found to benefit IAM dataset identification rates, but had

no effect on the scribal dataset (Section 4.5). This suggests that graphemes generated from medieval text are more consistent (rendering translation-invariance unnecessary); or that the position of the relative ink distribution within the grapheme 'window' is itself a writer-specific feature in medieval data.

In feature combinations, feature extraction using LDA provides the top identification performance on the IAM dataset for small codebooks, whereas PCA performs the most accurately on the scribal dataset for all codebook sizes (Section 5.3).

In classification, the style of the document of sample origin had a very strong, highly significant effect on the medieval dataset, reducing identification accuracy by as much as 22% when accounted for. In contrast, the IAM dataset reduction was only marginally significant, standing at approximately 3% (Section 6.1). This result demonstrates that the accuracy of the grapheme codebook method on scribal data is drawn in part from correctly matching samples based on a common document-style, rather than a writer-specific generalisation. This illustrates the impact and difficulty of working with the strong stylistic influences of medieval data.

Taken together, these results demonstrate that optimisations in data preparation, feature analysis, and classification are not equally applicable to modern and medieval data. Medieval writing requires domain-specific consideration for text normalisation and source-document handling. Unlike modern writing, it fails to benefit from translation-invariant grapheme comparison, LDA feature extraction, or additional ligature-based grapheme representation.

7.1.2 Cross-dataset results

For several of the experiments run, the techniques tested were robust to the variation in dataset type.

In grapheme formation, ligature-based grapheme segmentation performed poorly on both datasets, although this criterion was more informative on scribal than

modern data. At 61–72% and 44–53% identification accuracy on the medieval and IAM datasets respectively, ligature-based graphemes clearly contain a significant amount of writer-specific information, but never reaches the classification rates of character body-based segmentation (Section 4.3). Similarly, the complexity distance measure also performed poorly (but above chance-level) on both datasets, suggesting that non-spatial information carries few writer-specific signals (Section 4.5).

The top-performing techniques at the feature analysis stage are most robust to the application dataset. PCA feature extraction is a consistent top-performer on both datasets, increasing in accuracy with codebook size and exceeding baseline identification accuracy on 500-grapheme codebooks (Section 5.3). PCA-based feature selection (Section 5.4.1) is also the top-performing selection method tested, suggesting that spanning feature-vector variation regardless of writer-class considerations is the best feature analysis approach.

When comparing the codebook similarity selection approaches, the necessity of a broad distribution of grapheme similarities holds on both datasets. Minimising similarity between codebook graphemes also underperforms maximising similarity, confirming that codebooks composed of outlier graphemes are less writer-discriminating than those with small grapheme variations only, regardless of dataset type (Section 5.4.6).

At the verification stage both datasets perform similarly overall, with a small proportion of each dataset classifiable at a stringent threshold with very high accuracy. A consistent increase as the threshold varies concludes with a significant increase in the accuracy with which correct and incorrect labels can be marked, giving more information overall than a purely identification-based system (Section 6.2.4).

7.1.3 Proposed methods

The final contributions of this thesis are the following methods proposed over the course of these experiments to improve data analysis in writer identification.

Section 5.3.3 introduces the approach of combining the feature extraction coefficients calculated via PCA and LDA to compute the individually highest-weighted graphemes. This is used as both for visualisation of the extracted features in terms of the original component graphemes (Section 5.3.3), and as a basis for feature selection (Section 5.4.1).

Section 6.1.1 describes a classification strategy for quantifying the impact that document origin has on the identification of its constituent writing samples. This method can be used to analyse and adjust the expected identification accuracies for individual datasets, or to compare different features on the same dataset in terms of their ability to generalise over document-style distractions.

Finally, the work in Section 6.2 employs the precise distance information typically discarded in the nearest-neighbour classifier as the basis of a verification system. An algorithm is given to convert distances into confidence levels, and a MATLAB implementation is provided.

7.2 Limitations and Further Work

The broadest limitations of the work presented in this thesis are inherent in the data used: for the findings to apply beyond these specific datasets, the IAM and scribal datasets must be representative of the data available in their respective time periods. Continuing the experiments carried out in this thesis on alternative datasets could confirm this, with additional applications to medieval data being of particular interest. There are however several barriers to manuscript data availability, such as a lack of systematic digitisation, the requirement for organisation by writer at the page level, and frequent copyright issues in reproducing manuscripts held in collections.

The samples chosen to represent typical contemporary handwriting data were from the 100-writer task from the IAM handwriting dataset, a benchmark data subset marked specifically for writer identification experiments. Unfortunately during the course of this work the collection identifiers have been withdrawn, although the individual images it comprised are still available. The choice of this data was intended to be typical of the datasets on which feature development takes place, however, the difference in sample size compared to the medieval dataset is undesirable. Further work examining the effects of sample-size in each dataset are necessary to quantify the impact (if any) that this may have had on the results presented. A related issue is the variation in the number of writers between datasets, which may have affected e.g. the comparative stability of ICA feature extraction.

The medieval dataset is highly unbalanced in respect of the number of samples available per-writer and per-manuscript. Whilst more complex to handle in evaluation terms, this imbalance is typical of the data available and must be handled in developing and analysing scribal identification techniques.

The sample-size used most often (and retained in this work) is per-page of a given document. In historical texts, a complete manuscript may have been produced and illustrated by a number of scribes, who are likely to have imitated each other's writing styles to ensure continuity in the presentation. In producing a training set classification, writer classification therefore cannot be recorded per-document: it is likely required at a per-page level of granularity at a minimum, and possibly further refined for pages which are disputed or have a known switchover point – although these may also be left out of the training corpus entirely.

Variation in the sample-size used has not been examined as a variable in this work, but could be tested on a quantitative or qualitative basis, i.e. it could be enforced numerically (e.g. fixed number of graphemes allocated to each sample), or based on a writing-related unit such as page or paragraph. Quantitative approaches provide greater direct comparability of results, but may not reflect the context in which the writing was produced: for example, mixing graphemes from

different writing units may produce a different shape distribution than is typical of a particular text. Additionally (as the results in Chapter 6 demonstrate) subdividing a single text where the document style is a significant factor may overestimate the expected classification accuracy on an unrelated document. Qualitative approaches avoid these problems, but can produce samples with very varied text quantities. In a varied-content corpus a distinction such as 'paragraph' may also be impossible to apply consistently as it may not be equivalent or relevant to all documents (e.g. lists or recipes, pages with columns). Future work may consider how identification accuracy responds to changes in sample-size and division method. The minimum sample-size required to achieve a given level accuracy in each type of dataset is also unknown.

As the domain application of writer identification matures, it may be necessary to make use of the additional information in grayscale images for fine-tuning accuracy improvements. This area of research has additional challenges that distinguish it from contemporary writing. The writing instruments used to produce medieval script require little or no pen pressure, reducing the information available through this channel. As historical documents usually suffer from bleed-through and faded and degraded ink traces, care must be taken to minimise the effect this has on the grayscale information collected.

A limitation of the grapheme codebook feature itself is its reliance on a shape-based ink-trace representation: this may leave it unable to abstract writer-features well enough to generalise beyond the font style used in medieval scripts. Testing this hypothesis requires a dataset categorised by font as well as writer, with sufficient samples of each to draw conclusions. The document style factor tested (Section 6.1) is a first step towards this, but is an imperfect proxy for font as document style is linked strongly to period, location, and manuscript content type (e.g. prose, poetry, religious works) which were not controlled for.

Potential extensions to specific areas of work include examination of the possible interactions between grapheme segmentation and normalisation approaches,

and dynamic feature selection methods to produce specific target-codebook similarity distributions. A number of extensions to the verification framework outlined are also possible, such as a change of distance criterion (changing distance to closest match to e.g. ratio of distances to closest predicted match:non-match), or writer-specific distance thresholds. The latter however requires far more data per scribe than currently available.

7.3 Conclusion

The work presented in this thesis has demonstrated clear areas of domain-specific considerations when handling medieval manuscript data, as opposed to a conventional contemporary dataset. The experiments reported have also identified characteristics common to both datasets, and developed three novel methodological approaches to the analysis of writer identification data. Although further development is required to integrate this research into a viable software platform for use in the humanities, it is hoped that this thesis has made a useful contribution to the field.

Appendix A

Datasets

A.1 IAM

The IAM database (Marti and Bunke, 1999) is a collection of grayscale PNG images of varied-content text[1]. The images are available in full pages, or at the text line or word level, and are fully labelled. The content of the samples is freeform English text, divided into several subject categories: although this has a negligible effect for writer identification purposes, it is beneficial for other uses of the corpus. Each test page contains about a paragraph of text, using a separate guideline sheet to ensure that the lines are horizontal and well-spaced. The database currently consists of around 1500 samples, from roughly 650 writers. More details can be found in Marti and Bunke (1999).

[1] Available for download from http://www.iam.unibe.ch/fki/databases/iam-handwriting-database

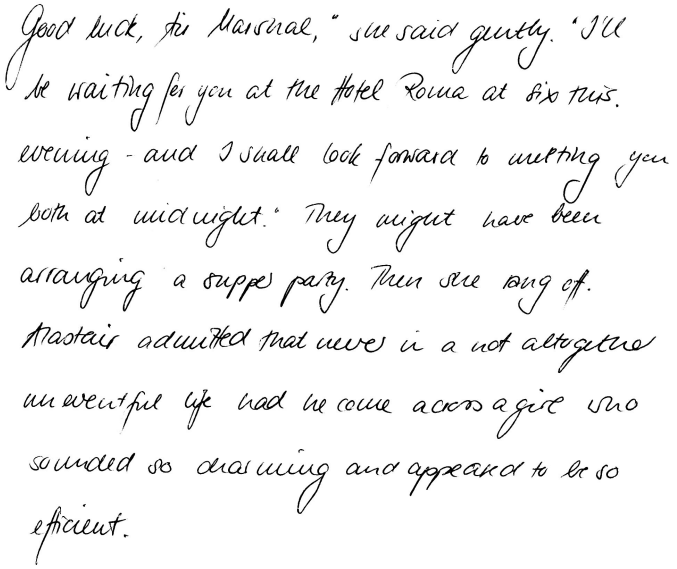


FIGURE A.1: Full page IAM dataset sample (Marti and Bunke, 2002)

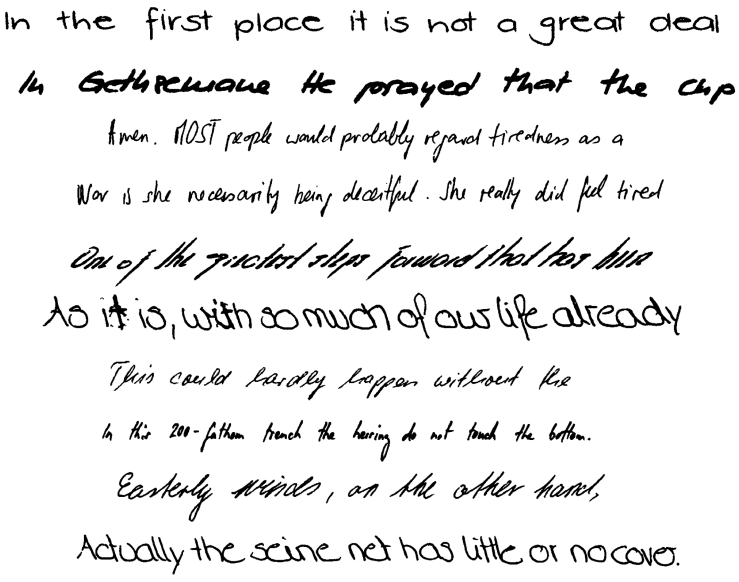
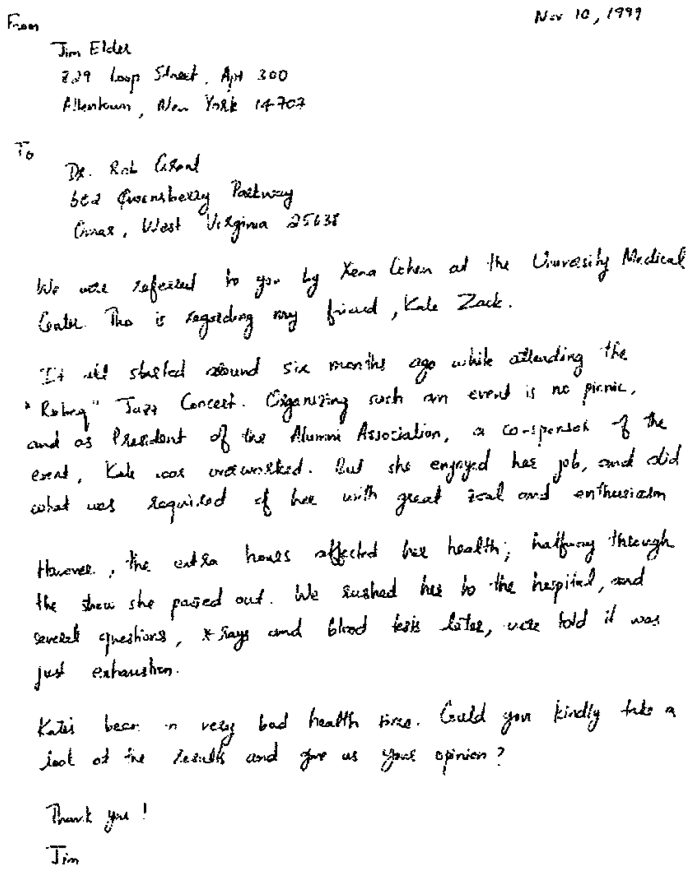


FIGURE A.2: Selected lines from IAM database handwriting samples (Marti and Bunke, 2002)


From

Nov 10, 1999

Jim Elder
739 Loop Street, Apt 300
Allentown, New York 14707

To

Dr. Rob Cleod
602 Gooseberry Parkway
Omar, West Virginia 25638

We were referred to you by Xena Cohen at the University Medical Center. She is regarding my friend, Kate Zack.

It all started around six months ago while attending the "Rainy" Jazz Concert. Organizing such an event is no picnic, and as President of the Alumni Association, a co-sponsor of the event, Kate was overworked. But she enjoyed her job, and did what was required of her with great zeal and enthusiasm

However, the extra hours affected her health; halfway through the show she passed out. We rushed her to the hospital, and several questions, X-rays and blood tests later, were told it was just exhaustion.

Kate's been in very bad health since. Could you kindly take a look at the records and give us your opinion?

Thank you!
Jim

FIGURE A.3: Complete CEDAR letter sample (Srihari et al., 2002)

A.2 CEDAR

The CEDAR database is a fixed-content dataset composed of roughly 1500 writers, each contributing three samples of the CEDAR letter. The writers were selected to be representative of the American population as stratified by many factors, including gender, handedness, country of origin, level of education and ethnicity. The sample text is in the form of a letter in English to be copied which has been designed to contain all the digits and capitals, and every lower-case character in word start, middle and end position, as this may affect the allograph used. Extensive manual segmentation of characters and some words has been carried out. The dataset is described in Srihari et al. (2002).

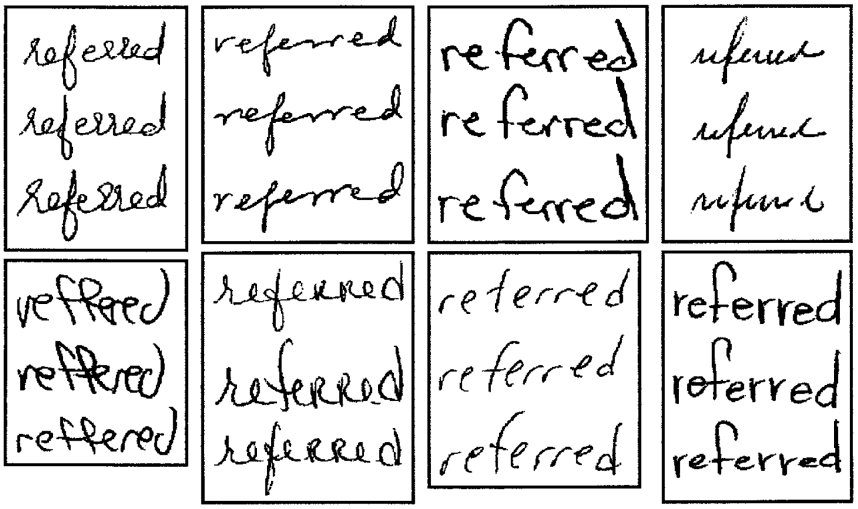


FIGURE A.4: Copies of the word 'referred' segmented from several different-writer samples of the CEDAR letter (Srihari et al., 2002)

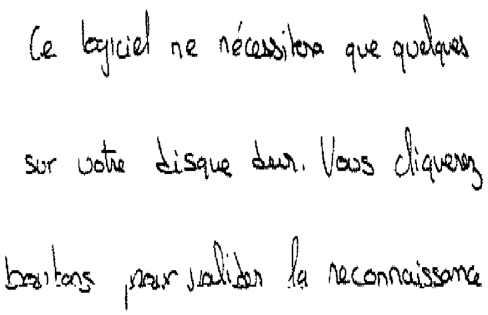


FIGURE A.5: Sample from a PSI database letter (Bensefia et al., 2005)

A.3 PSI

The PSI database consists of 88 samples of French text, one from each writer. Each sample is a copy of one of two possible fixed-text letters of 107 or 98 words each, producing mainly cursive written text (Bensefia et al., 2002).

FIGURE A.6: Further examples of the two letters from the PSI dataset (Bensefia et al., 2002)

A.4 Firemaker

The Firemaker database contains Dutch writing samples from 250 writers, each contributing 4 pages of a fixed format. One page is a copy of a fixed-content text, another is free form text of varied content generated by asking the writer to describe a cartoon and the third is a fixed text copied out in uppercase. The final page is a fixed text copied in disguised or 'forged' handwriting - the writers were asked to impersonate someone, but it is implied that no reference style was given, i.e. the writers were attempting to conceal their own natural style rather than specifically adopt someone else's (Schomaker et al., 2004). Guidelines were used that were removed by the digitisation process, and the paper and writing instruments were standardised (Bulacu et al., 2003).

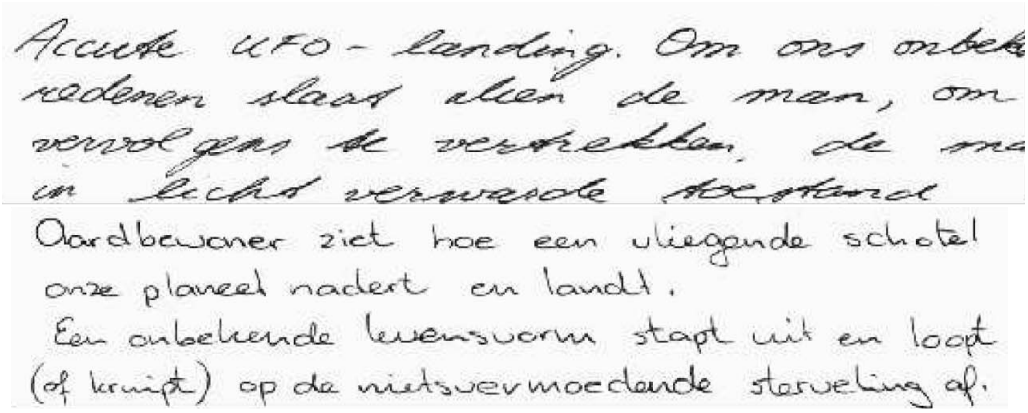
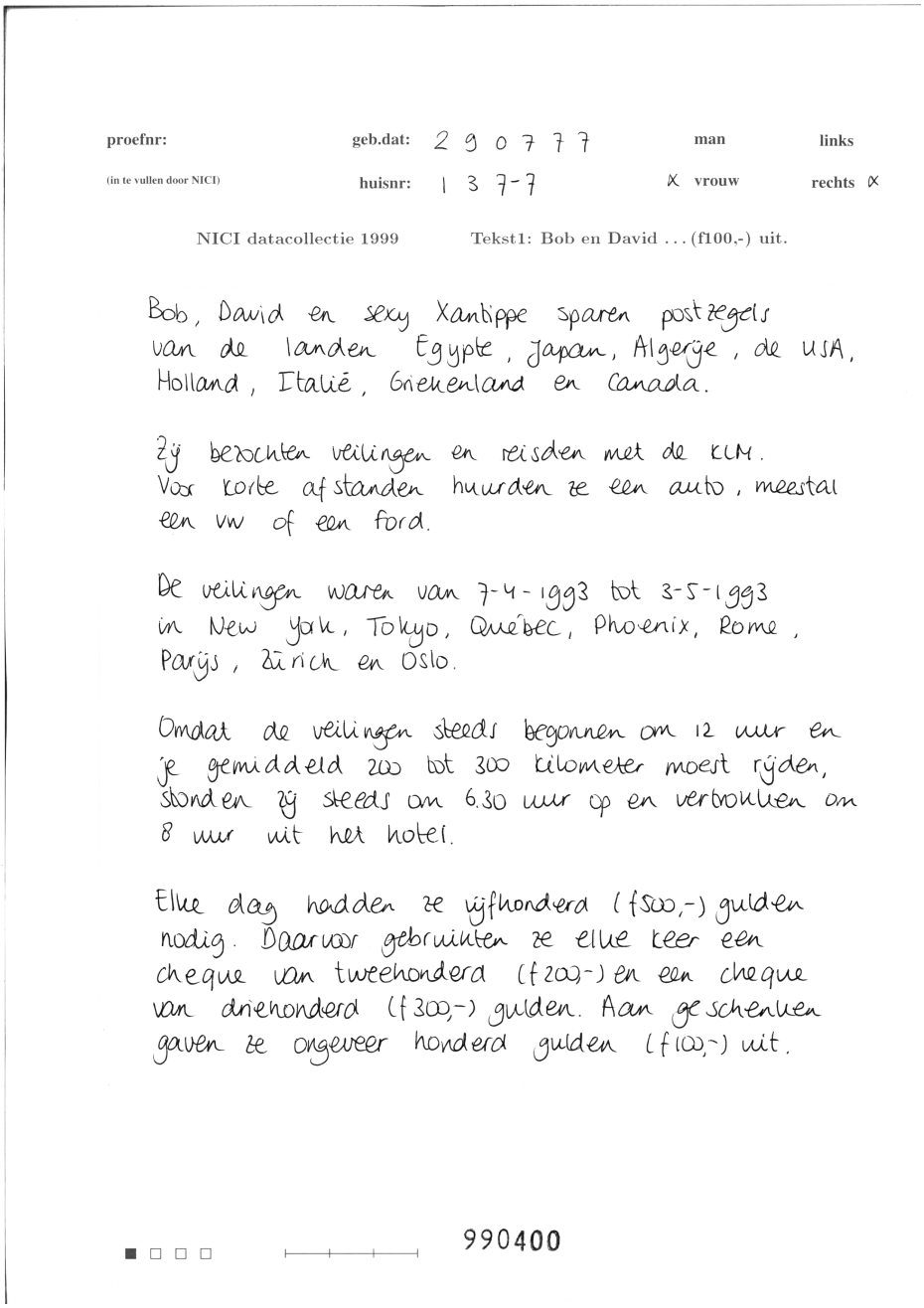


FIGURE A.7: Two samples of writing extracted from Firemaker database pages (Bulacu and Schomaker, 2007b)



proefnr: geb.dat: 2 9 0 7 7 7 man links

(in te vullen door NICI) huisnr: 1 3 7-7 X vrouw rechts X

NICI datacollectie 1999 Tekst1: Bob en David ... (f100,-) uit.

Bob, David en sexy Xantippe sparen postzegels van de landen Egypte, Japan, Algerije, de USA, Holland, Italië, Groenenland en Canada.

Zij bezochten veilingen en reisden met de KLM. Voor korte afstanden huurden ze een auto, meestal een vw of een ford.

De veilingen waren van 7-4-1993 tot 3-5-1993 in New York, Tokyo, Québec, Phoenix, Rome, Parijs, Zürich en Oslo.

Omdat de veilingen steeds begonnen om 12 uur en je gemiddeld 200 tot 300 kilometer moest rijden, stonden zij steeds om 6.30 uur op en vertrokken om 8 uur uit het hotel.

Elke dag hadden ze vijfhonderd (f500,-) gulden nodig. Daarvoor gebruikten ze elke keer een cheque van tweehondera (f200,-) en een cheque van drienonderd (f300,-) gulden. Aan geschenken gaven ze ongeveer honderd gulden (f100,-) uit.

990400

FIGURE A.8: Full-page sample of a Firemaker database text (van der Maaten, 2005)

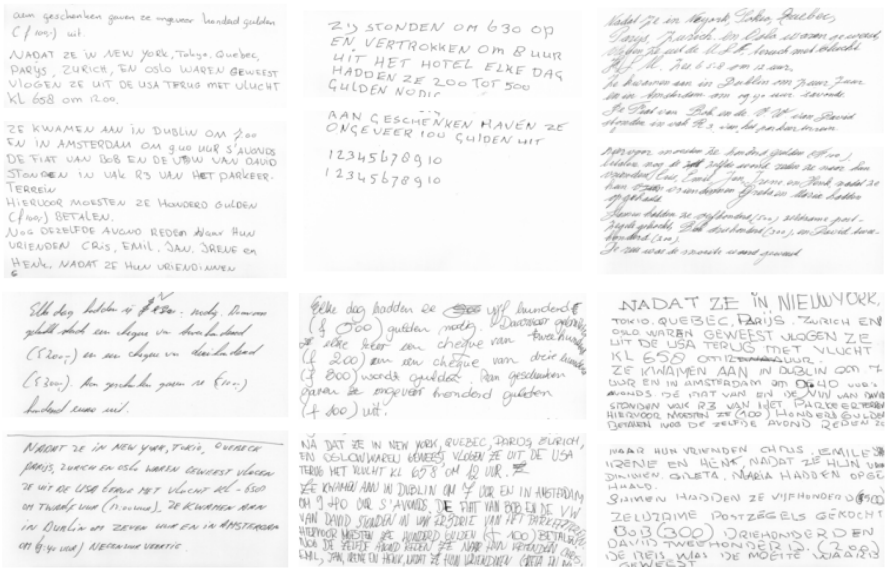


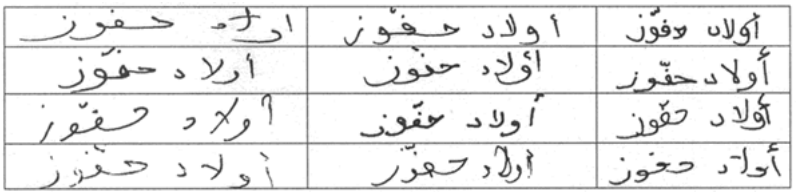
FIGURE A.9: Examples of writing from the NFI set, each split into two (Brink et al., 2007)

A.5 NFI

The NFI database has been collected by the Dutch National Forensic Institute, and consists of forms filled out by around 1300 suspects in criminal cases. There are usually 2 pages per person, with some exceptions, for a total of 3500 written forms containing a mixture of cursive and capitalised text. The majority of the content is a standardised text transcribed from dictation. An unusual feature of the NFI set is the untidiness of the data - the forms were not lined in any way, and collection was not carried out with automatic processing in mind. Some artifacts such as marks or holes are present, and the images would generally require preprocessing before use (Brink et al., 2007).

A.6 ImUnipen

The Unipen dataset is a database of online handwriting information, that is, it contains timing, velocity and coordinate data rather than static images, and it is



Label	Quantity	Shape	Label	Quantity	Shape
ا_EJ_B	789	لا	ح_MJ_B	363	حـ
ا_EJ_M	1066	ـلا	ح_Mم_MJ_B	64	حمـ
إ_EJ_B	120	لإ	ح_Mن_B	100	حنـ
أ_EJ_B	357	لأ	خ_MJ_B	304	خـ
ج_MJ_B	530	جـ	م_MJ_B	450	مـ

FIGURE A.10: Word and ligature images from the IFN/ENIT Arabic word database

generally used in online writer identification where this information is incorporated into the feature set (Guyon et al., 1994). However, static images can be derived from the online trace data, to be used in offline writer identification experiments. The resulting dataset of images generated from 215 writers of the Unipen set has been named the ImUnipen set. Details of the image generation methods can be found in Bulacu and Schomaker (2006).

A.7 IFN/ENIT

The IFN/ENIT database [2](Pechwitz et al., 2002) is a collection of handwritten Arabic town and village names, made available for the purposes of testing handwriting recognition and identification algorithms. The database content was selected with the aim of being similar to writing that might be found on a letter. Word images were collected from whole-page forms, segmented at the character/ligature level, binarised, and manually checked for writing errors. A significant amount of metadata is presented alongside, such as ground truth transcriptions, writer information (identifier, age, profession), and postcode.

[2]http://www.ifnenit.com/

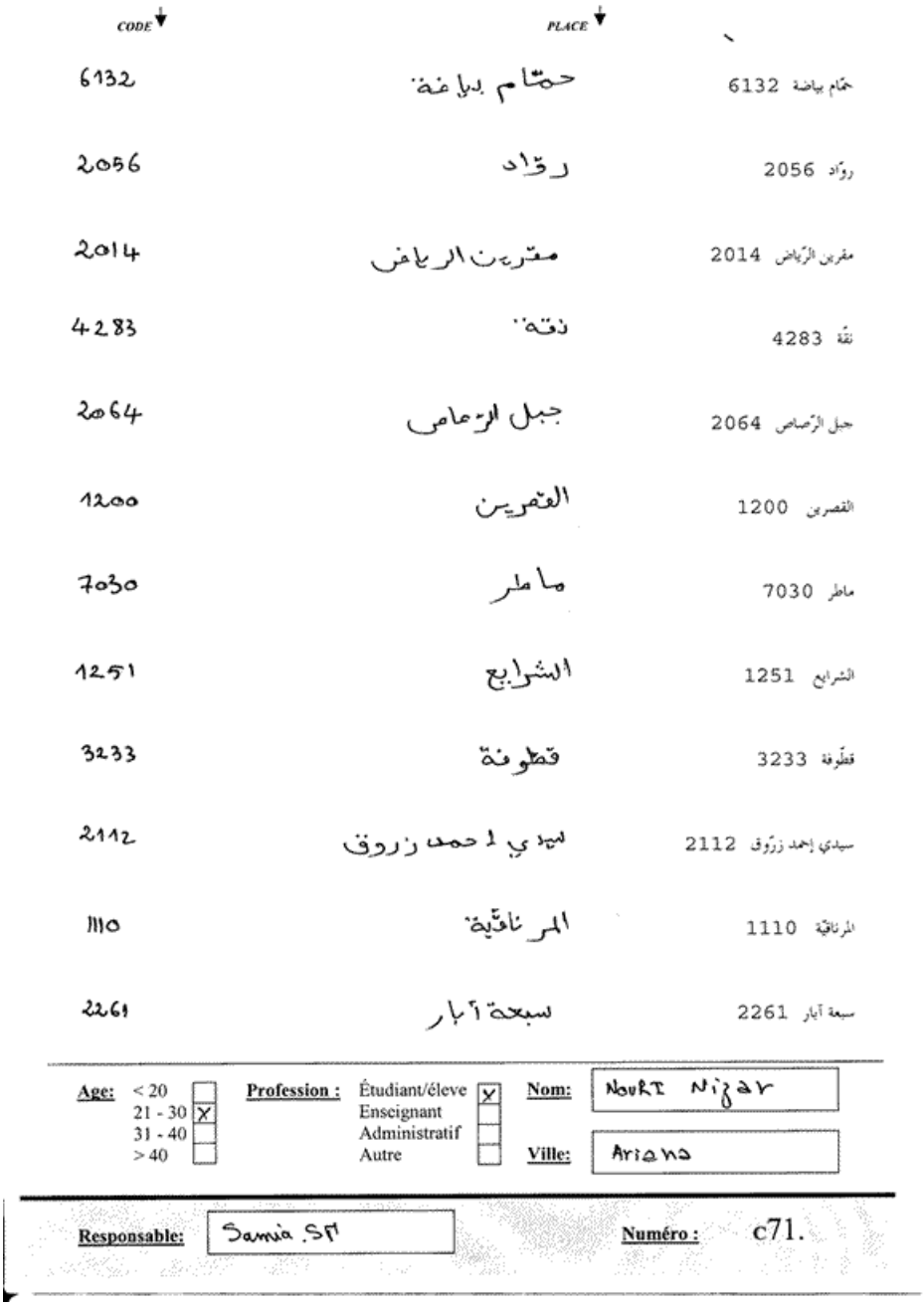


FIGURE A.11: Original whole-page form from the IFN/ENIT database showing postcodes and place names

Appendix B

Offline Writer Identification Feature Performance

The following table is a comprehensive listing of most of the work in individual writer identification feature extraction. Results are grouped by feature, and within these, in descending order of performance. Feature combinations, survey papers and repeated results are excluded. Multiple entries may appear for a single paper that test many features, although where many variations on a feature have been tested, representative results will be chosen. The number of writers listed are those from the test set, where train and test figures vary. In general, accuracy is expected to decrease with increasing numbers of writers to be distinguished, and to be noticeably higher for samples of fixed content rather than varied. Information on the most commonly used datasets is given in Appendix A. Other factors not listed that affect accuracy include the number of samples, sample size per writer, training data:test data ratio and the distance measure used, especially with nearest-neighbour classifiers. Top-1 hit rates are given to the nearest whole percent, and some have been estimated from graphs where no other figures are supplied.

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
slant distribution	250	Firemaker	varied	1-nn	48	Bulacu and Schomaker (2007b)
slant distribution	10	Medieval manuscripts	varied	1-nn	47	Bulacu and Schomaker (2007a)
slant distribution	650	IAM	varied	1-nn	46	Bulacu and Schomaker (2007b)
slant distribution	250	Firemaker (split lines)	varied	1-nn	45	Bulacu and Schomaker (2003)
slant distribution	900	Firemaker+IAM	varied	1-nn	43	Bulacu and Schomaker (2006)
slant distribution	250	Firemaker (uppercase)	fixed	1-nn	43	Bulacu and Schomaker (2007b)
slant distribution	250	Firemaker	varied	1-nn	39	van der Maaten (2005)
slant distribution	250	Firemaker	varied	1-nn	35	Bulacu et al. (2003)
slant distribution	150	Firemaker (uppercase)	fixed	1-nn	34	Schomaker et al. (2007)
slant distribution	251	Firemaker	unknown	1-nn	33	van Erp et al. (2003)
slant distribution	350	Arabic words	fixed	1-nn	31	Bulacu et al. (2007a)
slant distribution	250	Firemaker	varied	1-nn	26	Bulacu and Schomaker (2003)
differentiated slant distribution	250	Firemaker	varied	1-nn	45	Bulacu et al. (2003)
edge-hinge distribution	250	Firemaker (uppercase)	fixed	1-nn	84	Bulacu and Schomaker (2007b)
edge-hinge distribution	150	Firemaker (uppercase)	fixed	1-nn	83	Schomaker and Bulacu (2004)
edge-hinge distribution	350	Arabic words	fixed	1-nn	82	Bulacu et al. (2007a)
edge-hinge distribution	650	IAM	varied	1-nn	81	Bulacu and Schomaker (2007b)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
edge-hinge distribution	298	IAM (200 chars)	varied	1-nn	81	Brink et al. (2008)
edge-hinge distribution	250	Firemaker	varied	1-nn	81	Bulacu and Schomaker (2007b)
edge-hinge distribution	10	Medieval manuscripts	varied	1-nn	81	Bulacu and Schomaker (2007a)
edge-hinge distribution	900	Firemaker+IAM	varied	1-nn	80	Bulacu and Schomaker (2006)
edge-hinge distribution	250	Firemaker (split lines)	varied	1-nn	78	Bulacu and Schomaker (2003)
edge-hinge distribution	192	Firemaker (200 chars)	varied	1-nn	76	Brink et al. (2008)
edge-hinge distribution	250	Firemaker	varied	1-nn	72	van der Maaten (2005)
edge-hinge distribution	251	Firemaker	unknown	1-nn	71	van Erp et al. (2003)
edge-hinge distribution	250	Firemaker	varied	1-nn	63	Bulacu and Schomaker (2003)
edge-hinge combinations (3,7)	250	Firemaker	varied	1-nn	81	van der Maaten (2005)
edge-hinge combinations (3,7,9)	250	Firemaker	varied	1-nn	78	van der Maaten (2005)
edge-hinge combinations (3,5,7,9)	250	Firemaker	varied	1-nn	77	van der Maaten (2005)
hor. direction co-occurrence distr.	251	Firemaker	unknown	1-nn	76	van Erp et al. (2003)
hor. direction co-occurrence distr.	10	Medieval manuscripts	varied	1-nn	71	Bulacu and Schomaker (2007a)
hor. direction co-occurrence distr.	650	IAM	varied	1-nn	68	Bulacu and Schomaker (2007b)
hor. direction co-occurrence distr.	250	Firemaker	varied	1-nn	68	Bulacu and Schomaker (2007b)
hor. direction co-occurrence distr.	900	Firemaker+IAM	varied	1-nn	65	Bulacu and Schomaker (2006)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
hor. direction co-occurrence distr.	250	Firemaker (split lines)	varied	1-nn	64	Bulacu and Schomaker (2003)
hor. direction co-occurrence distr.	250	Firemaker	varied	1-nn	53	Bulacu and Schomaker (2003)
hor. direction co-occurrence distr.	250	Firemaker (uppercase)	fixed	1-nn	51	Bulacu and Schomaker (2007b)
hor. direction co-occurrence distr.	350	Arabic words	fixed	1-nn	38	Bulacu et al. (2007a)
ver. direction co-occurrence distr.	250	Firemaker	varied	1-nn	66	Bulacu and Schomaker (2007b)
ver. direction co-occurrence distr.	650	IAM	varied	1-nn	65	Bulacu and Schomaker (2007b)
ver. direction co-occurrence distr.	900	Firemaker+IAM	varied	1-nn	59	Bulacu and Schomaker (2006)
ver. direction co-occurrence distr.	10	Medieval manuscripts	varied	1-nn	56	Bulacu and Schomaker (2007a)
ver. direction co-occurrence distr.	350	Arabic words	fixed	1-nn	39	Bulacu et al. (2007a)
ver. direction co-occurrence distr.	250	Firemaker (uppercase)	fixed	1-nn	37	Bulacu and Schomaker (2007b)
grapheme distribution (2D SOFM)	180	Arabic text	varied	1-nn	100	Ghiasi and Safabakhsh (2010)
grapheme distribution (2D SOFM)	150	Firemaker	fixed	1-nn	93	Schomaker et al. (2004)
grapheme distribution	192	Firemaker (200 chars)	varied	1-nn	82	Brink et al. (2008)
grapheme distribution (k-means)	650	IAM	varied	1-nn	80	Bulacu and Schomaker (2007b)
grapheme distribution (random)	150	Firemaker	varied	1-nn	79	van der Maaten (2005)
grapheme distribution (1D SOFM)	150	ImUnipen	varied	1-nn	79	Bulacu and Schomaker (2005a)
grapheme distribution (2D SOFM)	250	Firemaker	varied	1-nn	78	Bulacu and Schomaker (2005a)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
grapheme distribution (k-means)	150	ImUnipen	varied	1-nn	78	Bulacu and Schomaker (2005a)
grapheme distribution (k-means)	900	Firemaker+IAM	varied	1-nn	76	Bulacu and Schomaker (2006)
grapheme distribution (2D SOFM)	150	ImUnipen	varied	1-nn	76	Bulacu and Schomaker (2005a)
grapheme distribution (k-means)	250	Firemaker	varied	1-nn	75	Bulacu and Schomaker (2005a)
grapheme distribution (1D SOFM)	250	Firemaker	varied	1-nn	75	Bulacu and Schomaker (2005a)
grapheme distribution (k-means)	10	Medieval manuscripts	varied	1-nn	73	Bulacu and Schomaker (2007a)
grapheme distribution (2D SOFM)	150	Firemaker (uppercase)	fixed	1-nn	72	Schomaker et al. (2007)
grapheme distribution	298	IAM (200 chars)	varied	1-nn	70	Brink et al. (2008)
grapheme distribution (k-means)	250	Firemaker (uppercase)	fixed	1-nn	65	Bulacu and Schomaker (2007b)
grapheme distribution (2D SOFM)	150	Firemaker (uppercase)	fixed	1-nn	63	Schomaker et al. (2004)
grapheme distribution (2D SOFM)	150	Firemaker	varied	1-nn	62	Schomaker et al. (2004)
grapheme distribution (k-means)	350	Arabic words	fixed	1-nn	61	Bulacu et al. (2007a)
grapheme distribution (2D SOFM)	150	Firemaker (forged)	fixed	1-nn	47	Schomaker et al. (2004)
hor. run-length distribution (bg)	10	Medieval manuscripts	varied	1-nn	33	Bulacu and Schomaker (2007a)
hor. run-length distribution (bg)	192	Firemaker (200 chars)	varied	1-nn	23	Brink et al. (2008)
hor. run-length distribution (bg)	250	Firemaker	varied	1-nn	22	Bulacu et al. (2003)
hor. run-length distribution (bg)	298	IAM (200 chars)	varied	1-nn	21	Brink et al. (2008)
hor. run-length distribution (bg)	250	Firemaker	varied	1-nn	26	van der Maaten (2005)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
hor. run-length distribution (bg)	150	Firemaker (uppercase)	fixed	1-nn	26	Schomaker et al. (2007)
hor. run-length distribution (bg)	251	Firemaker	unknown	1-nn	20	van Erp et al. (2003)
hor. run-length distribution (bg)	250	Firemaker	varied	1-nn	18	Bulacu and Schomaker (2007b)
hor. run-length distribution (bg)	250	Firemaker	varied	1-nn	13	Bulacu and Schomaker (2003)
hor. run-length distribution (ink)	250	Firemaker	varied	1-nn	11	Bulacu et al. (2003)
hor. run-length distribution (bg)	650	IAM	varied	1-nn	10	Bulacu and Schomaker (2007b)
hor. run-length distribution (bg)	250	Firemaker (split lines)	varied	1-nn	9	Bulacu and Schomaker (2003)
hor. run-length distribution (bg)	900	Firemaker+IAM	varied	1-nn	8	Bulacu and Schomaker (2006)
hor. run-length distribution (bg)	250	Firemaker (uppercase)	fixed	1-nn	8	Bulacu and Schomaker (2007b)
hor. run-length distribution (bg)	350	Arabic words	fixed	1-nn	3	Bulacu et al. (2007a)
ver. run-length distribution (bg)	10	Medieval manuscripts	varied	1-nn	44	Bulacu and Schomaker (2007a)
ver. run-length distribution (bg)	250	Firemaker	varied	1-nn	29	van der Maaten (2005)
ver. run-length distribution (bg)	251	Firemaker	unknown	1-nn	27	van Erp et al. (2003)
ver. run-length distribution (bg)	150	Firemaker (uppercase)	fixed	1-nn	21	Schomaker et al. (2007)
ver. run-length distribution (bg)	250	Firemaker	varied	1-nn	18	Bulacu et al. (2003)
ver. run-length distribution (bg)	250	Firemaker	varied	1-nn	16	Bulacu and Schomaker (2007b)
ver. run-length distribution (ink)	250	Firemaker	varied	1-nn	12	Bulacu et al. (2003)
ver. run-length distribution (bg)	900	Firemaker+IAM	varied	1-nn	10	Bulacu and Schomaker (2006)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
ver. run-length distribution (bg)	250	Firemaker (uppercase)	fixed	1-nn	9	Bulacu and Schomaker (2007b)
ver. run-length distribution (bg)	650	IAM	varied	1-nn	8	Bulacu and Schomaker (2007b)
ver. run-length distribution (bg)	350	Arabic words	fixed	1-nn	3	Bulacu et al. (2007a)
brush/ink density distribution	150	Firemaker (uppercase)	fixed	1-nn	69	Schomaker et al. (2007)
brush/ink density distribution	250	Firemaker (split lines)	varied	1-nn	62	Bulacu and Schomaker (2003)
brush	298	IAM (200 chars)	varied	1-nn	57	Brink et al. (2008)
brush/ink density distribution	250	Firemaker	varied	1-nn	53	Bulacu and Schomaker (2003)
brush/ink density distribution	251	Firemaker	unknown	1-nn	53	van Erp et al. (2003)
brush	192	Firemaker (200 chars)	varied	1-nn	41	Brink et al. (2008)
brush/ink density distribution	250	Firemaker	varied	1-nn	30	van der Maaten (2005)
writer invariants	88	PSI	fixed (1 of 2)	1-nn	98	Bensefia et al. (2002)
writer invariants	88	PSI	fixed (1 of 2)	1-nn	93	Bensefia et al. (2005)
writer invariants	88	PSI	fixed (1 of 2)	1-nn	93	Bensefia et al. (2003)
writer invariants	150	IAM	varied	1-nn	87	Bensefia et al. (2005)
writer invariants	39	French historical	varied	1-nn	72	Bensefia et al. (2003)
stroke-fragment reference base	50	IAM	varied	1-nn	94	Siddiqi and Vincent (2007)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
stroke-fragment reference base	20	French texts	varied	1-nn	¿85	Seropian et al. (2003)
stroke-fragment reference base	50	IAM	varied	1-nn	84	Siddiqi and Vincent (2007)
Hermite filters	5	IAM+historical texts	probably varied	SVM	98	Imdad et al. (2007)
Gabor filters	17	Chinese characters	probably fixed	1-nn	96	Zhu et al. (2000)
Gabor filters + PCA/ICA	55	Chinese text	fixed	5-nn	93	Ubul et al. (2009)
Gabor-energy features	25	Arabic text	fixed	1-nn	88	Shahabi and Rahmati (2006)
Hermite filters	30	IAM+historical texts	probably varied	SVM	83	Imdad et al. (2007)
Fourier transform of Gabor output	25	Arabic text	fixed	1-nn	80	Shahabi and Rahmati (2006)
sigmoidal transform of Gabor filters	25	Arabic text	fixed	1-nn	80	Shahabi and Rahmati (2006)
wavelet-based GGD	10	Chinese text	varied	1-nn	80	He et al. (2005)
2D Gabor filters	20	probably English text	varied	1-nn	76	Said et al. (1998)
2D Gabor filters	10	Chinese text	varied	1-nn	70	He et al. (2005)
Gabor filters	20	music sheets	varied	5-nn	64	Fornéz et al. (2009)
Gabor energy features	40	Arabic text	probably fixed	1-nn	56	Shahabi and Rahmati (2009)
symmetric Gabor filters	25	Arabic text	fixed	1-nn	36	Shahabi and Rahmati (2006)
Fourier transform of Gabor output	40	Arabic text	probably fixed	1-nn	49	Shahabi and Rahmati (2009)
Gabor filter	11	Medieval manuscripts	varied	1-nn	30	Amos (2004)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
Villasenor (2) wavelet	150	Firemaker (uppercase)	fixed	1-nn	15	Schomaker et al. (2007)
Daubechies (14) wavelet	150	Firemaker (uppercase)	fixed	1-nn	15	Schomaker et al. (2007)
Odegard wavelet	150	Firemaker (uppercase)	fixed	1-nn	14	Schomaker et al. (2007)
Adelson wavelet	150	Firemaker (uppercase)	fixed	1-nn	14	Schomaker and Bulacu (2004)
Antonini wavelet	150	Firemaker (uppercase)	fixed	1-nn	14	Schomaker and Bulacu (2004)
Brislawn wavelet	150	Firemaker (uppercase)	fixed	1-nn	14	Schomaker and Bulacu (2004)
Haar wavelet	150	Firemaker (uppercase)	fixed	1-nn	5	Schomaker et al. (2007)
grayscale co-occurrence matrix	20	music sheets	varied	5-nn	66	Fornéz et al. (2009)
grayscale co-occurrence matrix	25	Arabic text	fixed	1-nn	64	Shahabi and Rahmati (2006)
grayscale co-occurrence matrix	20	English text	probably varied	1-nn	60	Said et al. (1998)
grayscale co-occurrence matrix	40	Arabic text	probably fixed	1-nn	31	Shahabi and Rahmati (2009)
hor. autocorrelation	150	Firemaker (uppercase)	fixed	1-nn	25	Schomaker et al. (2007)
hor. autocorrelation	250	Firemaker	varied	1-nn	16	Bulacu and Schomaker (2007b)
hor. autocorrelation	250	Firemaker (uppercase)	fixed	1-nn	16	Bulacu and Schomaker (2007b)
hor. autocorrelation	650	IAM	varied	1-nn	13	Bulacu and Schomaker (2007b)
hor. autocorrelation	251	Firemaker	unknown	1-nn	12	van Erp et al. (2003)
hor. autocorrelation	250	Firemaker	varied	1-nn	12	Bulacu et al. (2003)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
hor. autocorrelation	900	Firemaker+IAM	varied	1-nn	12	Bulacu and Schomaker (2007b)
hor. autocorrelation	250	Firemaker	varied	1-nn	0	van der Maaten (2005)
parameters of GMM	100	IAM	varied	1-nn	98	Schlapbach and Bunke (2006a)
OCR accuracy likelihood	100	IAM	varied	1-nn	97	Schlapbach and Bunke (2007b)
OCR accuracy likelihood	50	IAM	varied	1-nn	97	Schlapbach and Bunke (2007b)
OCR accuracy likelihood	100	IAM	varied	1-nn	97	Schlapbach and Bunke (2004b)
OCR accuracy likelihood	50	IAM	varied	1-nn	94	Schlapbach and Bunke (2004a)
fractal+geometric measures	20	IAM (single lines)	varied	neural net.	90	Marti et al. (2001)
fractal+geometric measures	20	IAM (single lines)	varied	1-nn	84	Marti et al. (2001)
fractal geometry transform	20	IAM (single lines)	varied	5-nn	93	Hertel and Bunke (2003)
fractal geometry transform	50	IAM (single lines)	varied	5-nn	84	Hertel and Bunke (2003)
statistics of lower+upper contours	20	IAM (single lines)	varied	5-nn	76	Hertel and Bunke (2003)
statistics of lower+upper contours	50	IAM (single lines)	varied	5-nn	53	Hertel and Bunke (2003)
statistics of connected-components	20	IAM (single lines)	varied	5-nn	54	Hertel and Bunke (2003)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
statistics of connected-components	50	IAM (single lines)	varied	5-nn	32	Hertel and Bunke (2003)
statistics of enclosed (loop) regions	20	IAM (single lines)	varied	5-nn	36	Hertel and Bunke (2003)
statistics of enclosed (loop) regions	50	IAM (single lines)	varied	5-nn	18	Hertel and Bunke (2003)
GSC (upper+lower)	875	CEDAR (characters)	fixed	1-nn	98	Zhang et al. (2003)
GSC (chars. of ‘referred’+‘b’+‘h’)	10	CEDAR (characters)	fixed	1-nn	97	Srihari et al. (2002)
GSC (chars. of ‘referred’+‘b’+‘h’)	100	CEDAR (characters)	fixed	1-nn	90	Srihari et al. (2002)
GSC (chars. of ‘referred’+‘b’+‘h’)	900	CEDAR (characters)	fixed	1-nn	82	Srihari et al. (2002)
GSC (digits 0-9)	875	CEDAR (digits)	fixed	1-nn	75	Zhang et al. (2003)
GSC (digits 0-5)	776	CEDAR (digits)	fixed	1-nn	58	Srihari et al. (2003)
GSC (word ‘referred’)	875	CEDAR (word)	fixed	1-nn	49	Zhang and Srihari (2003)
GSC (word ‘Medical’)	875	CEDAR (word)	fixed	1-nn	47	Zhang and Srihari (2003)
GSC (word ‘Cohen’)	875	CEDAR (word)	fixed	1-nn	44	Zhang and Srihari (2003)
GSC (word ‘been’)	875	CEDAR (word)	fixed	1-nn	40	Zhang and Srihari (2003)
GSC (digit ‘4’)	776	CEDAR (digits)	fixed	1-nn	16	Srihari et al. (2003)
GSC (digit ‘7’)	776	CEDAR (digits)	fixed	1-nn	12	Srihari et al. (2003)
GSC (digit ‘1’)	776	CEDAR (digits)	fixed	1-nn	4	Srihari et al. (2003)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
structural features of ‘th’	165	CEDAR (characters)	fixed	DistAl	67	Pervouchine and Leedham (2006)
structural features of ‘d’+‘y’+‘f’+‘th’	200	CEDAR (characters)	fixed	DistAl	58	Pervouchine and Leedham (2007)
structural features of ‘p’+‘y’	11	Medieval manuscripts	varied	Bayes	30	Maclean (2004)
statistical features of 3 chars.	34	Hebrew manuscripts	varied	linear Bayes	100	Bar-Yosef et al. (2007)
moments and ver. projection statistics	20	Greek words	fixed	neural net.	100	Zois and Anastassopoulos (1996)
gradient features	20	text pages	unknown	1-nn	100	Tapiador and Sigüenza (2004)
word image PCA components	40	Chinese words	fixed	1-nn	98	Zuo et al. (2002)
gradient features	50	Arabic text	varied	neural net.	94	Sadeghi ram and Moghaddam (2009)
directional element features	25	Chinese characters	fixed	1-nn	94	Wang et al. (2003)
stroke measurement distributions	40	IFN/ENIT	varied	Borda ranking	93	Abdi et al. (2009)
Zipf curve statistics	-	Italian medieval texts	varied	1-nn	80	Pareti and Vincent (2006)
autoregressive coefficients	422	French + Bengali text	varied	1-nn	62	Garain and Paquet (2009)
grayscale ink section statistics	-	no dataset information	unknown	LDA	51	Wirotius et al. (2003)
vertical char. structure ratios	11	Medieval manuscripts	varied	neural net.	40	Dibben (2004)
grayscale ink section statistics	-	no dataset information	unknown	1-nn	28	Wirotius et al. (2003)
statistics of ink and structure ratios	11	Medieval manuscripts	varied	1-nn	10	Fallon (2004)
entropy	250	Firemaker	varied	1-nn	3	Bulacu et al. (2003)
Hough features	250	Firemaker	varied	1-nn	2	van der Maaten (2005)

Feature	#Writers	Dataset	Content Type	Classifier	Top-1 (%)	Reference
fractal dimension	250	Firemaker	varied	1-nn	1	van der Maaten (2005)

Appendix C

Algorithms and Code Excerpts

This Appendix collects the algorithms and code excerpts referenced in the thesis, organised by chapter.

Chapter 3: Data Processing and Experimental Methodology

Algorithm 1 Connected-component extraction from ink pixel coordinates

```
allComponents = []

for pixel in coords:

  adjacentComponents = []

  for comp in allComponents:
    if comp.isAdjacentTo(pixel):
      adjacentComponents.add(comp)

  if adjacentComponents.size() != 0:
    allComponents.removeAll(adjacentComponents)
    mergedComponent = mergeAll(adjacentComponents, pixel)
    allComponents.add(mergedComponent)

  else:
    allComponents.add(pixel)
```

Algorithm 2 MATLAB implementation of the Euclidean or correlation distance

```
% Images are aligned by the top left-hand corner
% Images of different sizes will always differ
function dist = simpleImageDistance(im1, im2)

  % size() is given in total pixels
  avgpx = (im1.size() + im2.size()) / 2;

  % note any size differences between images
  excess = abs(im1.size() - im2.size()) / 2;

  % take image differences from overlapping areas only
  w = min(im1.width, im2.width);
  h = min(im1.height, im2.height);

  % overlap area started from top left corner
  i1 = im1(1:h, 1:w);
  i2 = im2(1:h, 1:w);

  imdiff = sum(sum(abs(i1 - i2)));

  % pixels present in one image but not the
  % other always compare different
  totaldiff = imdiff + excess;

  % normalise by mean image size
  dist = totaldiff/avgpx;
end
```

Algorithm 3 MATLAB implementation of the complexity of an image

```
function comp = complexity(img)
  w = img.width();
  h = img.height();

  rowChanges = 0;
  for y = 1:(h-1)
    changes = sum(abs(img(y, :) - img(y+1, :)));
    rowChanges = rowChanges + changes;
  end

  colChanges = 0;
  for x = 1:(w-1)
    changes = sum(abs(img(:, x) - img(:, x+1)));
    colChanges = colChanges + changes;
  end

  maxChanges = (w*(h-1) + h*(w-1));
  comp = (rowChanges + colChanges)/maxChanges;
end
```

Chapter 6: Classification

Algorithm 4 MATLAB implementation of label confidence calculation from nearest-neighbour distances

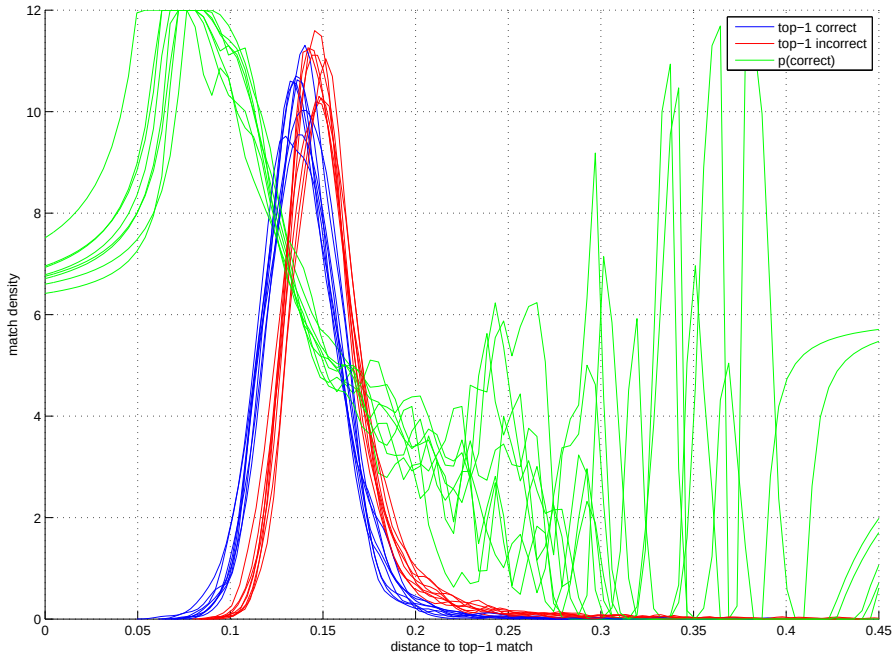
```
function [p, x] = pCorrect(training, test)
  [cys, cxs, iys, ixs] = calcConfidence(training);
  x = test.dists(1);

  % handle x-range edge cases
  xrangeC = and(x >= cxs(1), x <= cxs(end));
  xrangeIC = and(x >= ixs(1), x <= ixs(end));
  delta = 0.0001; % a fixed 'almost sure' error rate

  % edge 1: x in xrange of 'correct' distribution only
  if and(xrangeC, not(xrangeIC))
    p = 1 - delta;
    return;
  end

  % edge 2: x in xrange of 'incorrect' distribution only
  if and(xrangeIC, not(xrangeC))
    p = 0 + delta;
    return;
  end

  % edge 3: x in neither xrange: define test confidence
  % in proportion to the closest endpoints of each xrange
  if and(not(xrangeIC), not(xrangeC))
    distToFirstCx = min([cxs(1) cxs(end)] - x);
    distToFirstICx = min([ixs(1) ixs(end)] - x);
    p = 1-distToFirstCx/(distToFirstCx + distToFirstICx);
    return;
  end

  % edge cases not hit, proceed with normal interpolation:
  cy = interp1(cxs, cys, x);
  iy = interp1(ixs, iys, x);

  p = cy/(cy+iy);
end
```

Appendix D

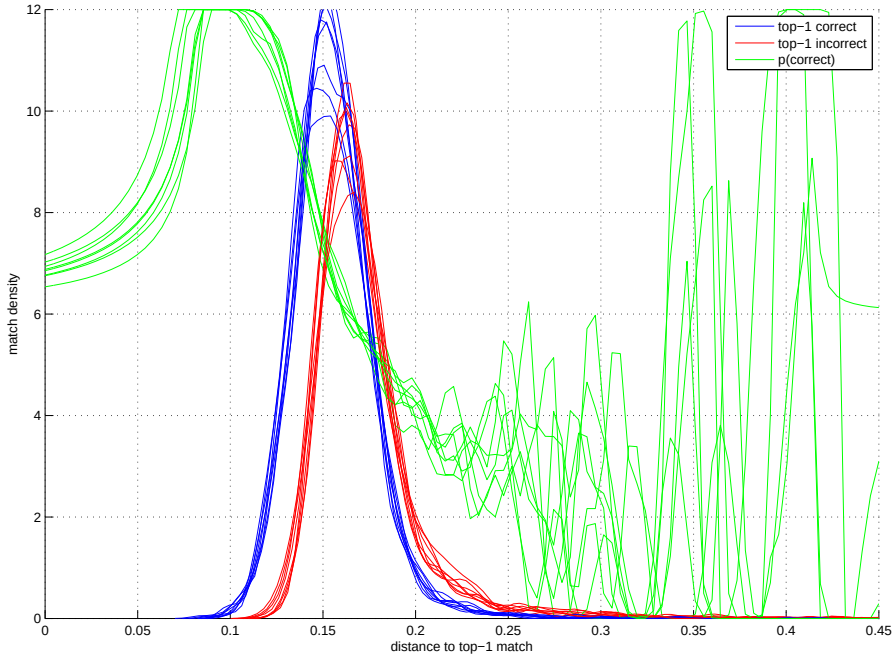
Verification results

Section 6.2 presents the design and implementation of a verification system based on the nearest-neighbour classifier. Identification accuracy results were calculated for a range of confidence thresholds for each dataset on codebook sizes of 50–500 graphemes. These are broadly similar and are thus omitted from the main text, but are included here for completeness.

D.1 IAM dataset verification results

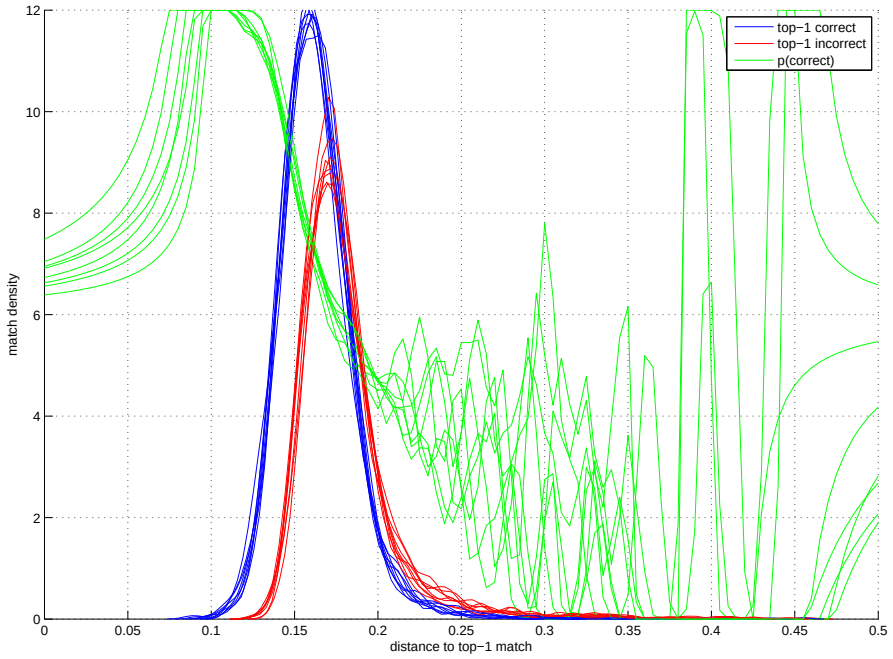


(A) Distance distributions and prediction confidence for codebook size 50

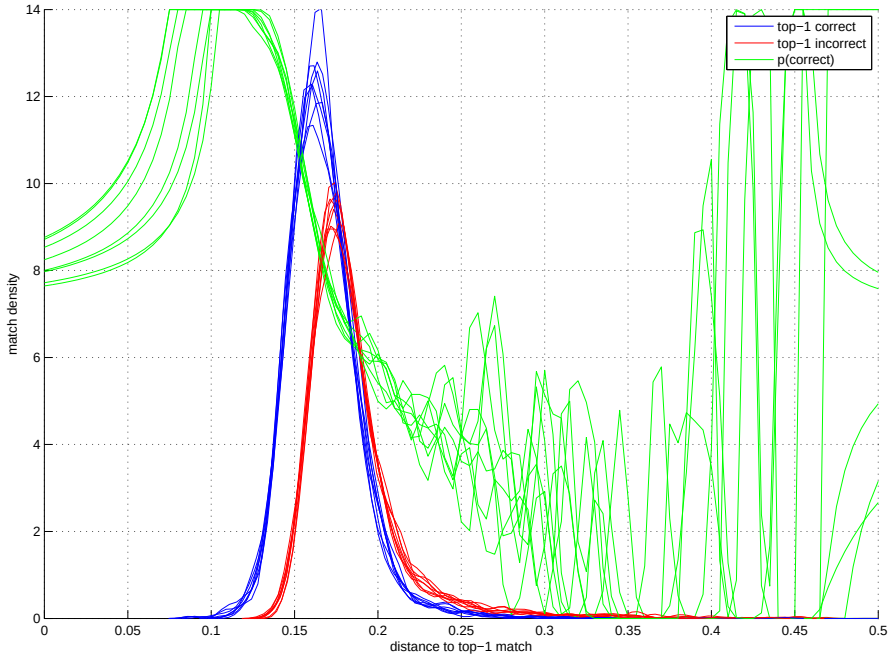


(B) Distance distributions and prediction confidence for codebook size 100

FIGURE D.1: Correct/Incorrect nearest-neighbour distance distributions for each dataset with prediction confidence, codebook sizes 50-100, eight runs

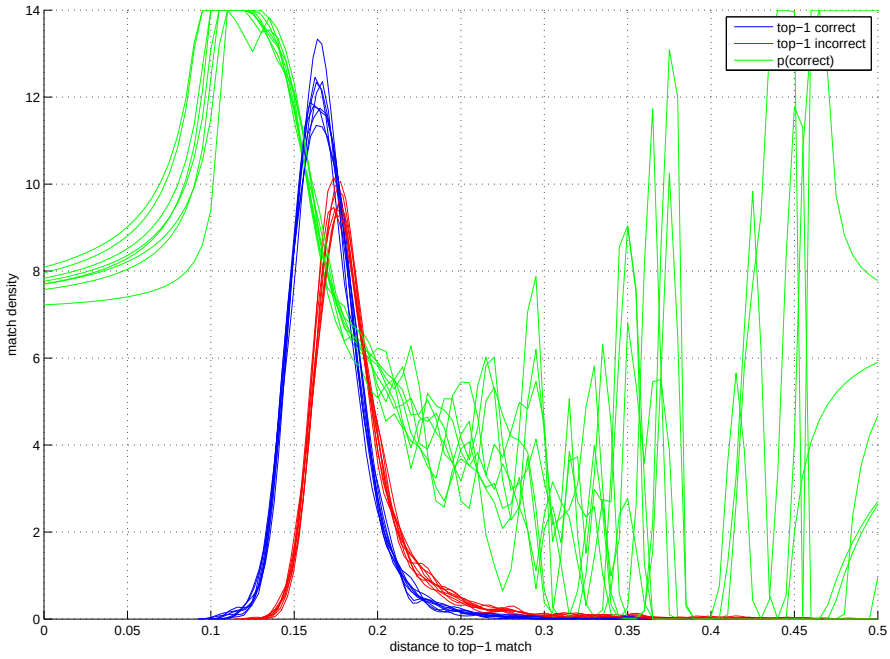


(A) Distance distributions and prediction confidence for codebook size 150

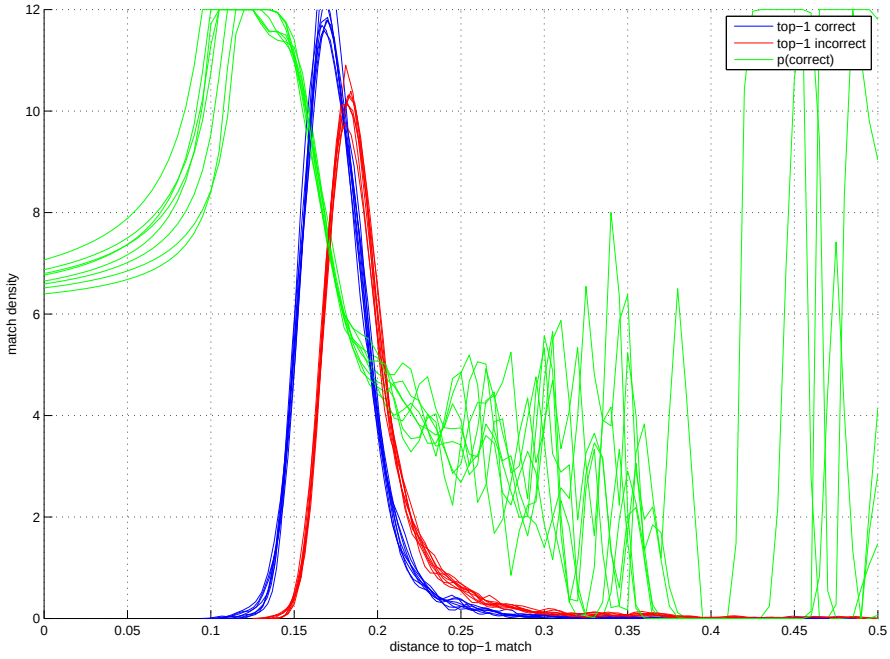


(B) Distance distributions and prediction confidence for codebook size 200

FIGURE D.2: Correct/Incorrect nearest-neighbour distance distributions for each dataset with prediction confidence, codebook sizes 150-200, eight runs

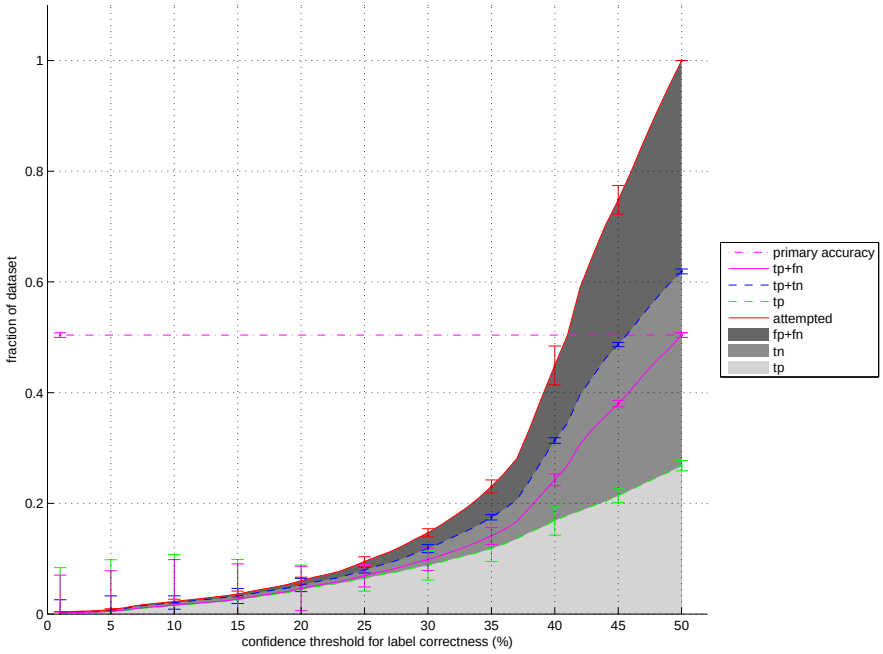


(A) Distance distributions and prediction confidence for codebook size 250

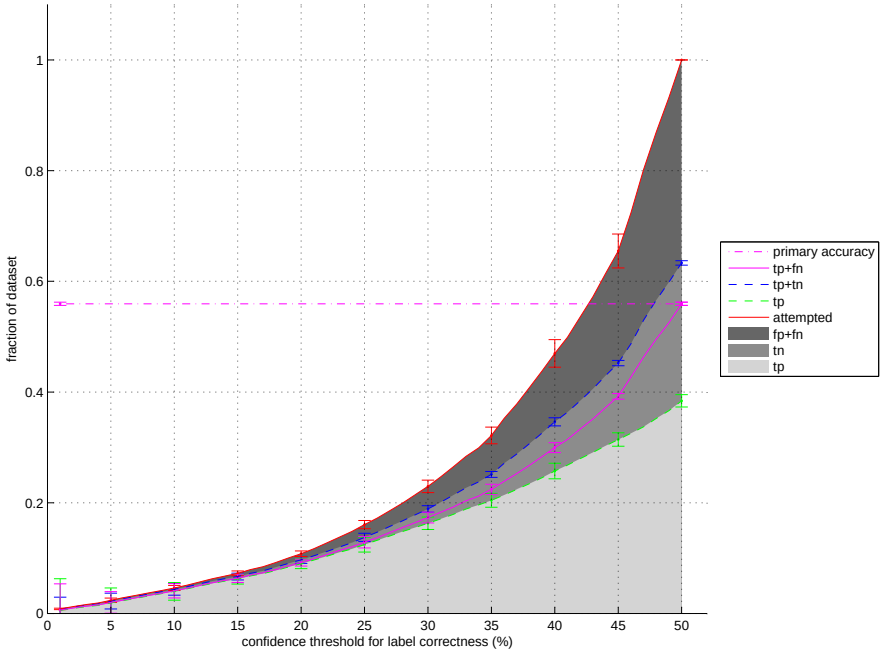


(B) Distance distributions and prediction confidence for codebook size 500

FIGURE D.3: Correct/Incorrect nearest-neighbour distance distributions for each dataset with prediction confidence, codebook sizes 250-500, eight runs

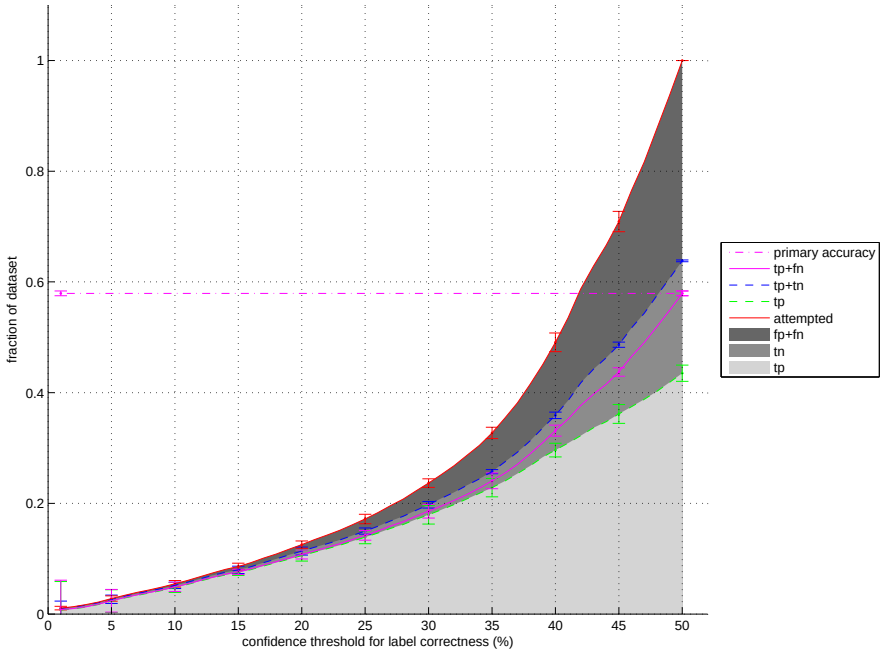


(A) Error/accuracy rates for codebook size 50

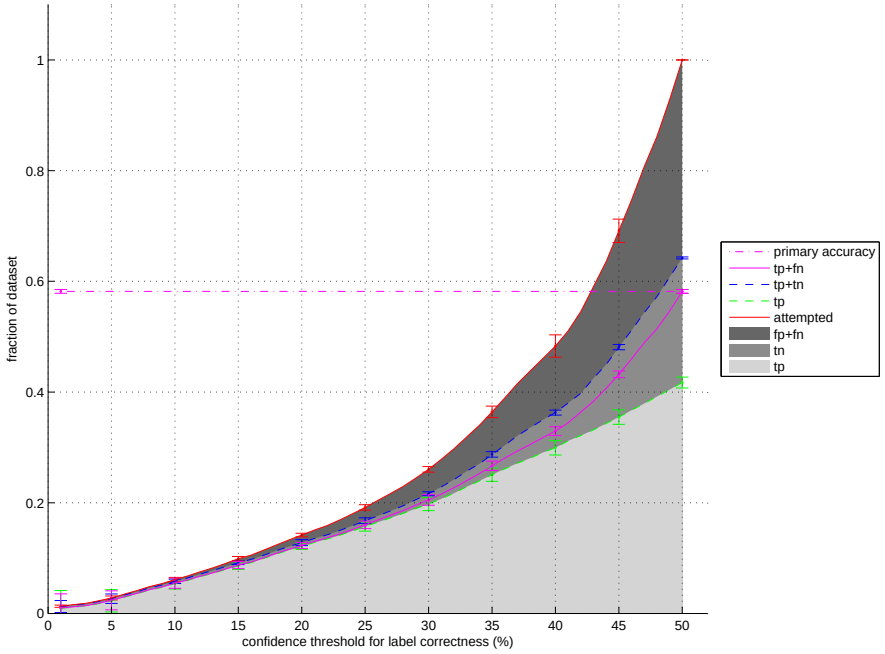


(B) Error/accuracy rates for codebook size 100

FIGURE D.4: Error rates for each confidence threshold as a proportion of the total dataset, codebook sizes 50-100, 8 runs mean and standard error

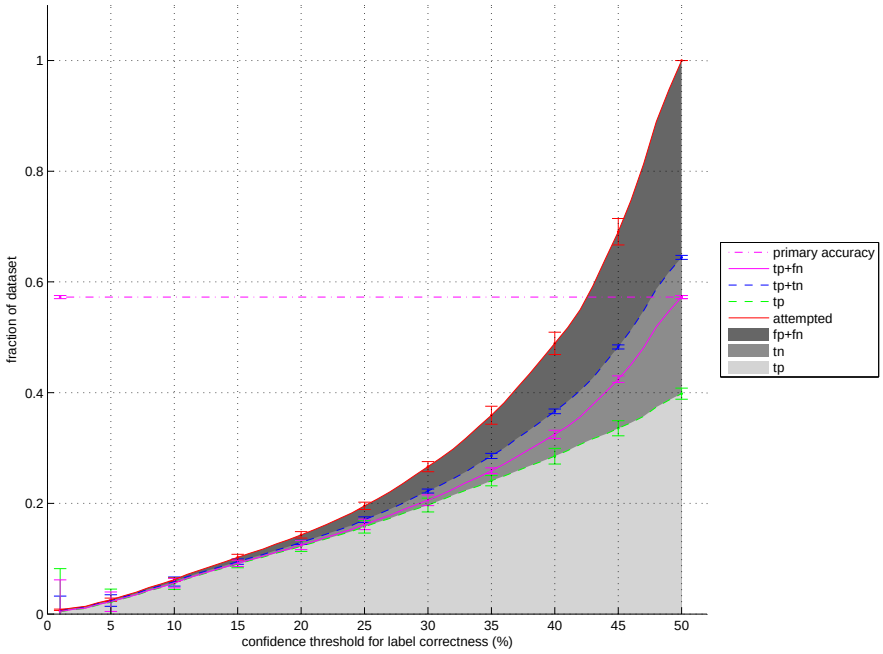


(A) Error/accuracy rates for codebook size 150

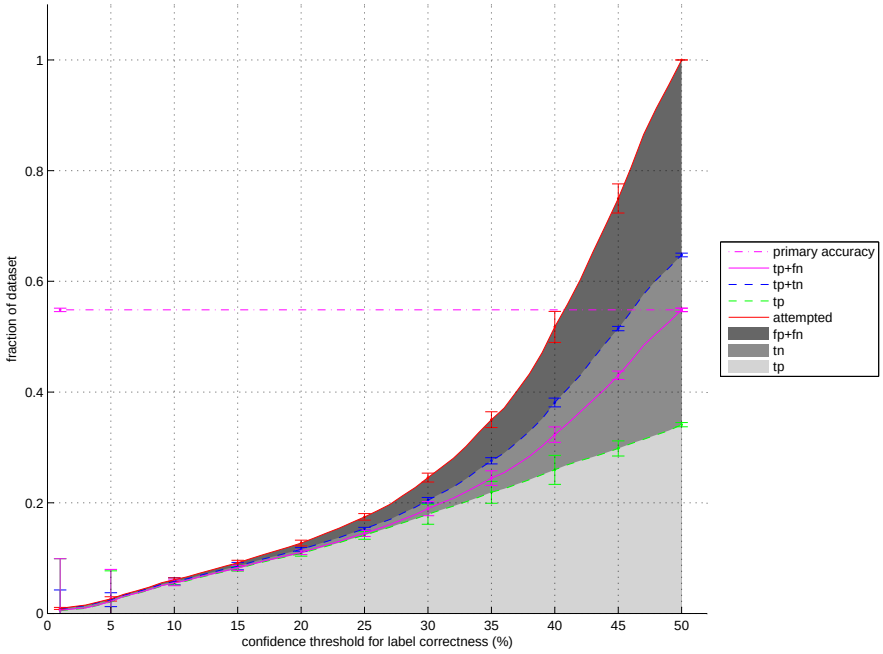


(B) Error/accuracy rates for codebook size 200

FIGURE D.5: Error rates for each confidence threshold as a proportion of the total dataset, codebook sizes 150-200, 8 runs mean and standard error

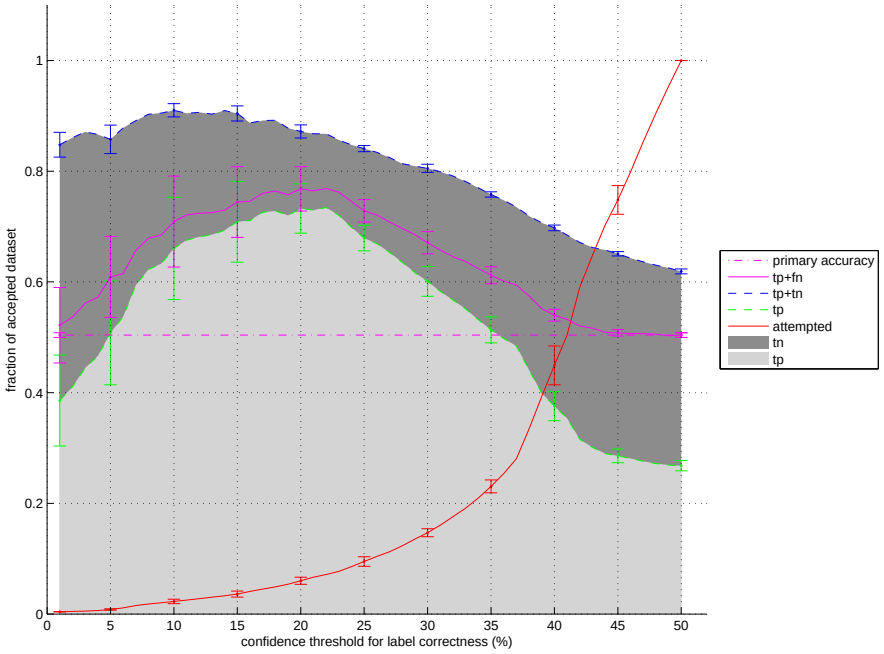


(A) Error/accuracy rates for codebook size 250

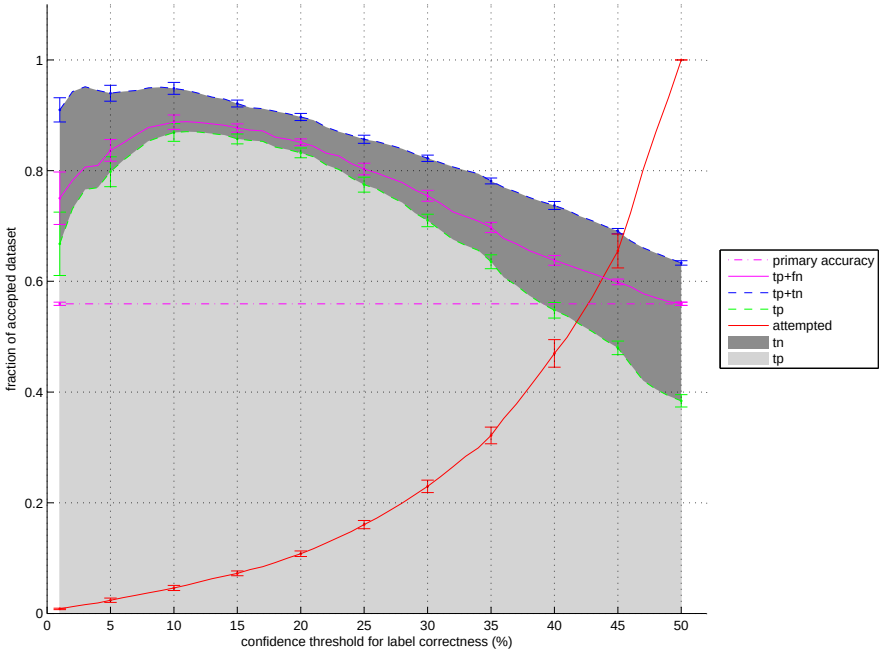


(B) Error/accuracy rates for codebook size 500

FIGURE D.6: Error rates for each confidence threshold as a proportion of the total dataset, codebook sizes 250-500, 8 runs mean and standard error

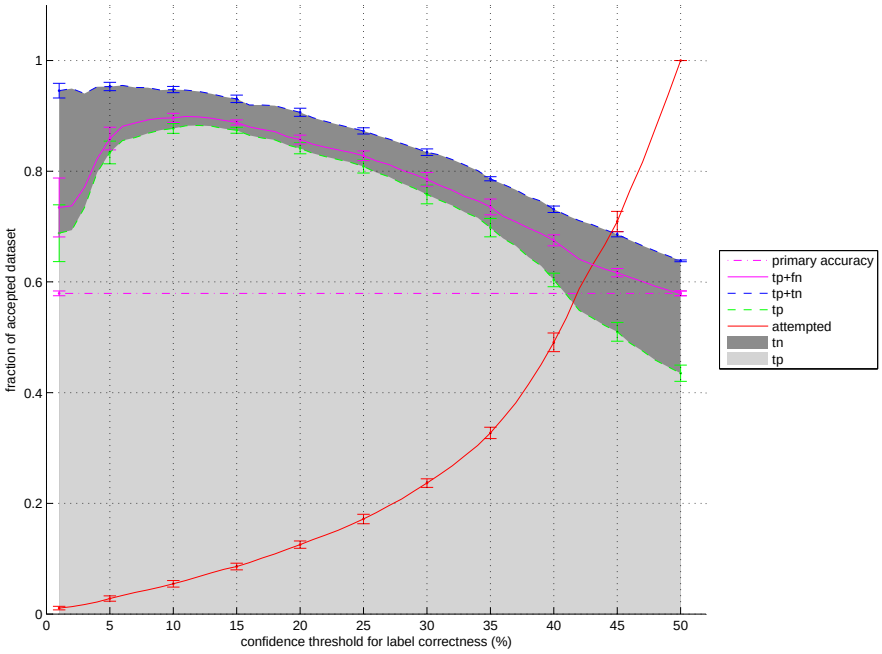


(A) Error/accuracy rates for codebook size 50

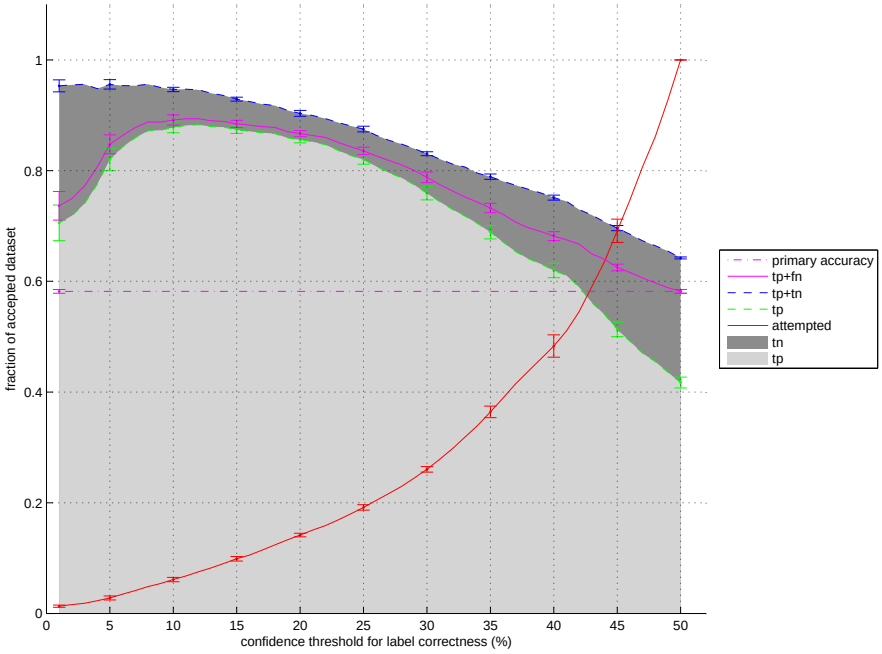


(B) Error/accuracy rates for codebook size 100

FIGURE D.7: Error rates for each confidence threshold as a proportion of the attempted dataset, codebook sizes 50-100, 8 runs mean and standard error

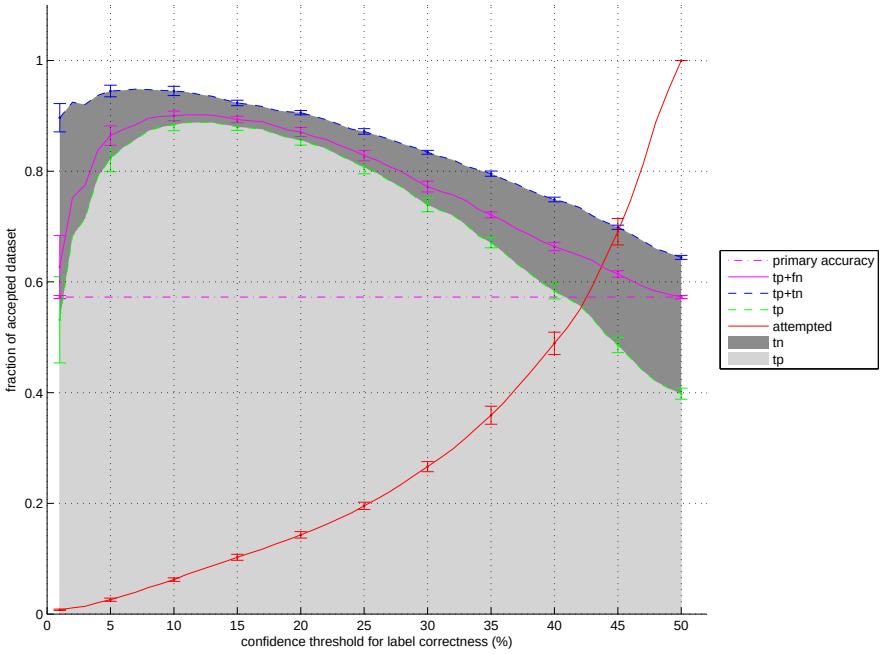


(A) Error/accuracy rates for codebook size 150

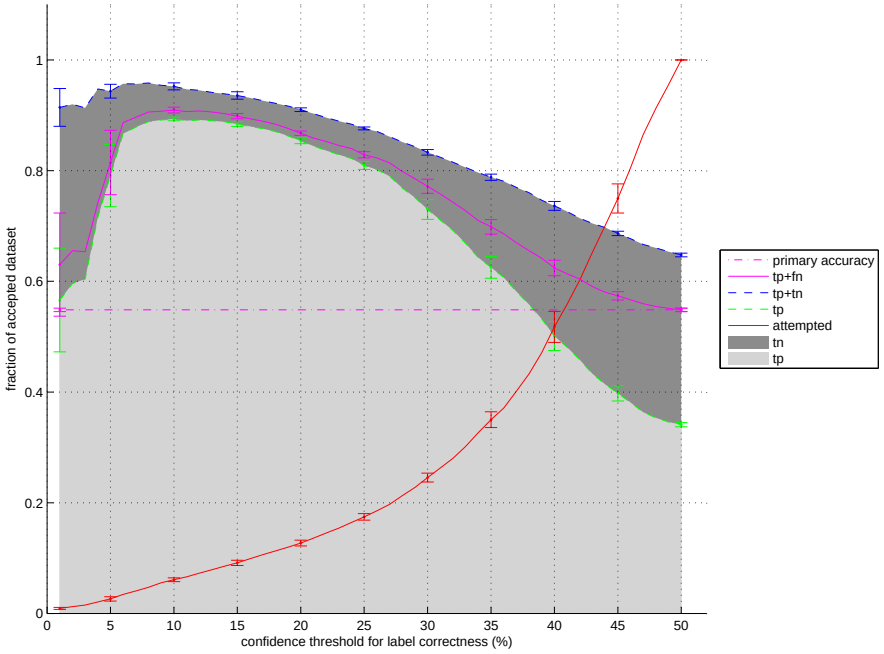


(B) Error/accuracy rates for codebook size 200

FIGURE D.8: Error rates for each confidence threshold as a proportion of the attempted dataset, codebook sizes 150-200, 8 runs mean and standard error



(A) Error/accuracy rates for codebook size 250

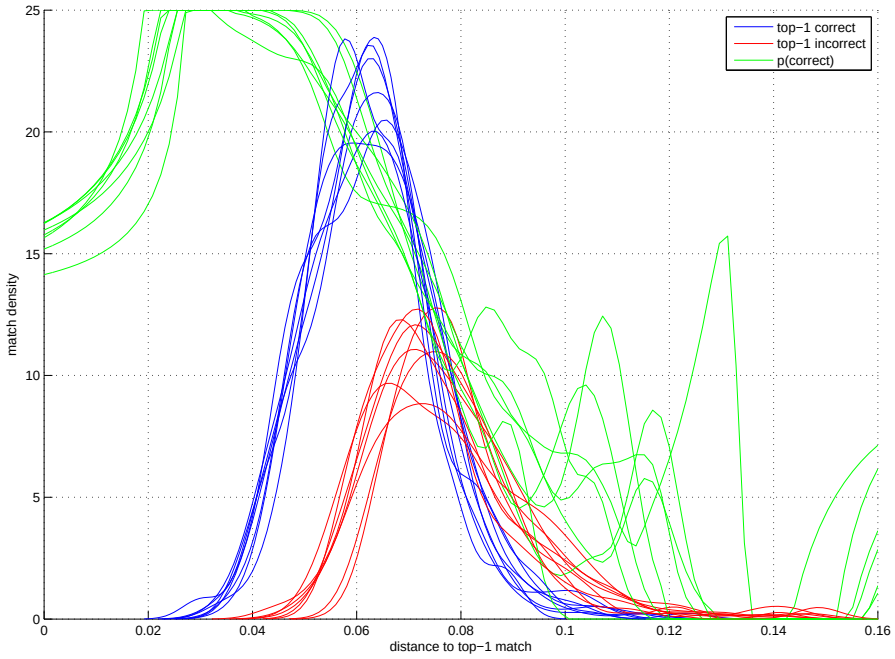


(B) Error/accuracy rates for codebook size 500

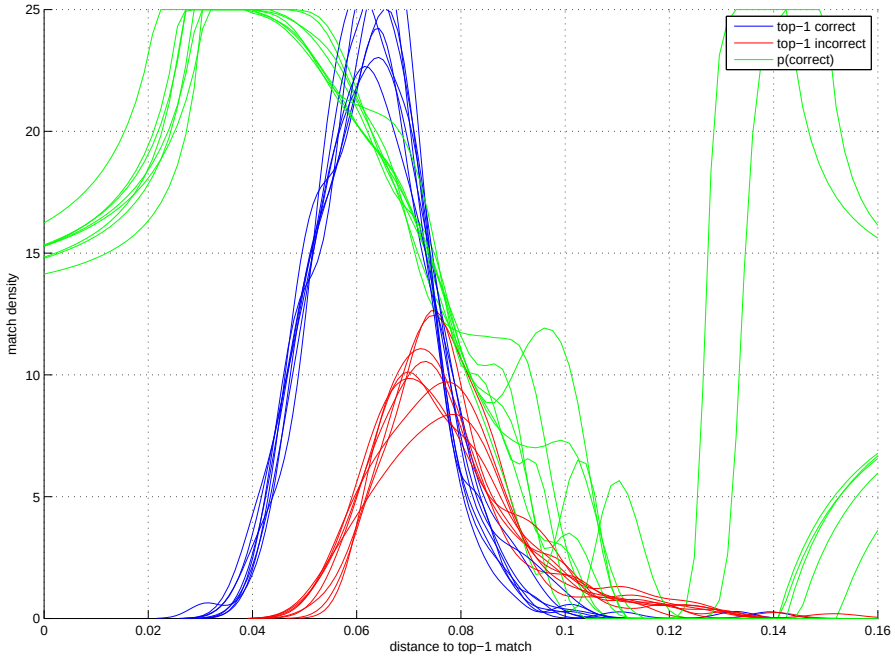
FIGURE D.9: Error rates for each confidence threshold as a proportion of the attempted dataset, codebook sizes 250-500, 8 runs mean and standard error

D.2 Scribal dataset verification results

In the following Figures, the equivalent scribal dataset results are given for codebook sizes $50 - 500$ for a range of confidence thresholds. As before, these results show the distributions of distances between the test sample and its nearest-neighbour for samples eventually identified correctly and incorrectly; and the error rates for the final verification system as a proportion of the dataset as a whole, and as a proportion of just the attempted samples.

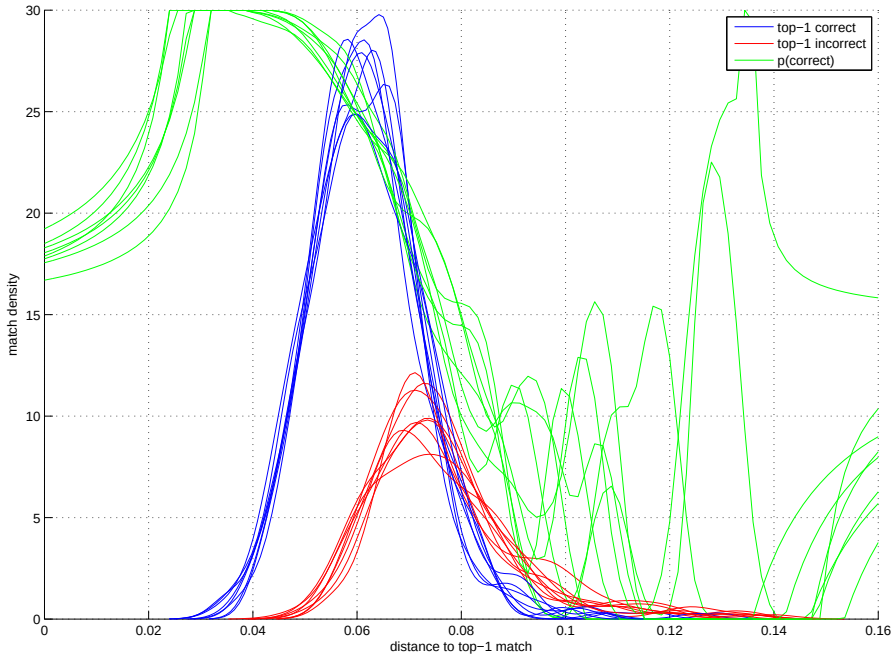


(A) Distance distributions and prediction confidence for codebook size 50

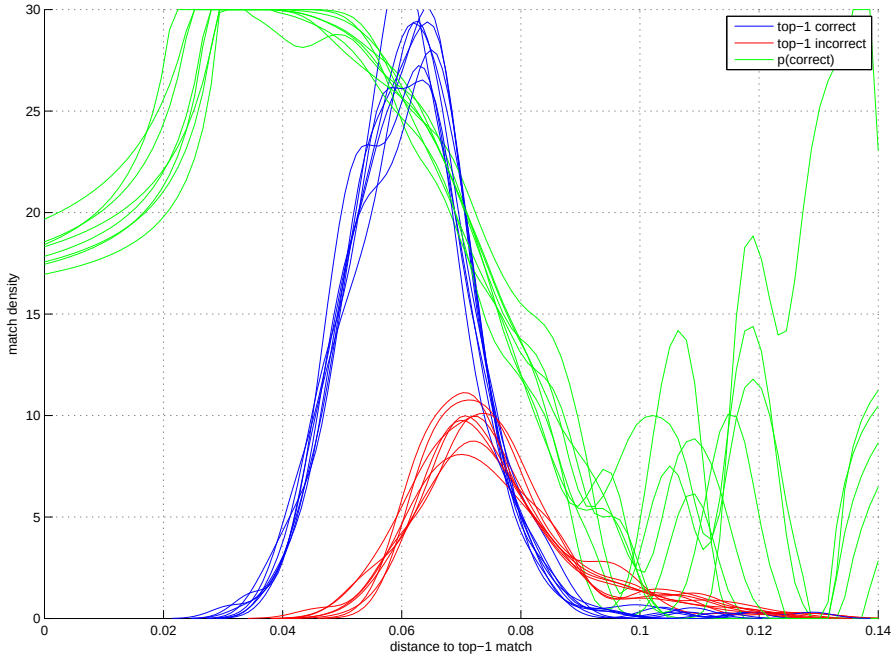


(B) Distance distributions and prediction confidence for codebook size 100

FIGURE D.10: Correct/Incorrect nearest-neighbour distance distributions for each dataset with prediction confidence, codebook sizes 50-100, eight runs

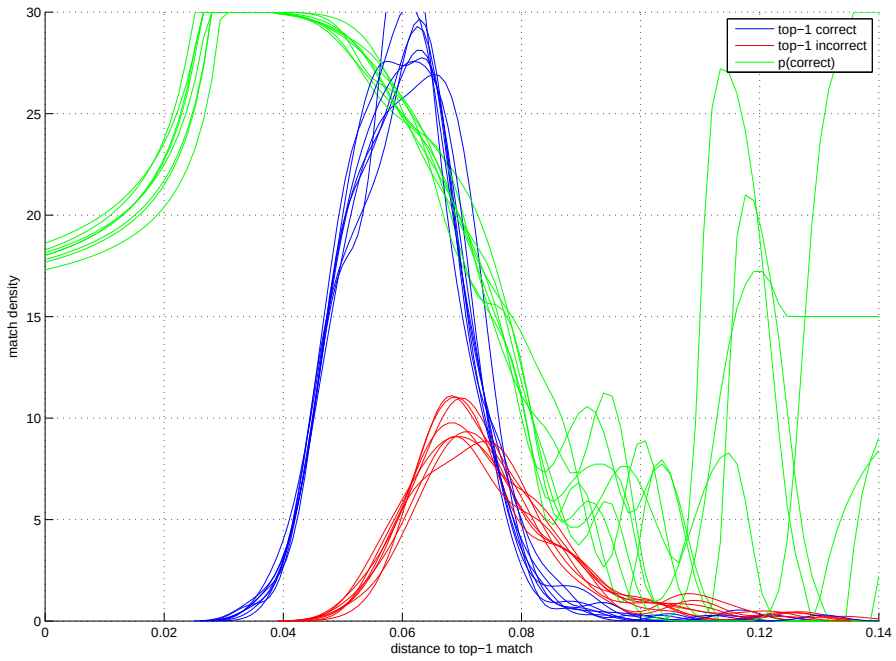


(A) Distance distributions and prediction confidence for codebook size 150

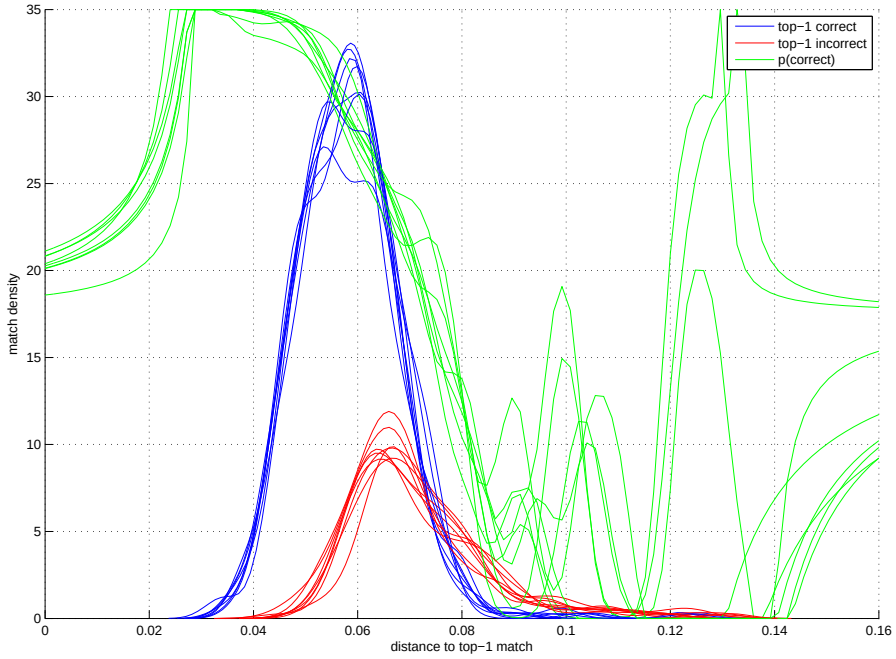


(B) Distance distributions and prediction confidence for codebook size 200

FIGURE D.11: Correct/Incorrect nearest-neighbour distance distributions for each dataset with prediction confidence, codebook sizes 150-200, eight runs

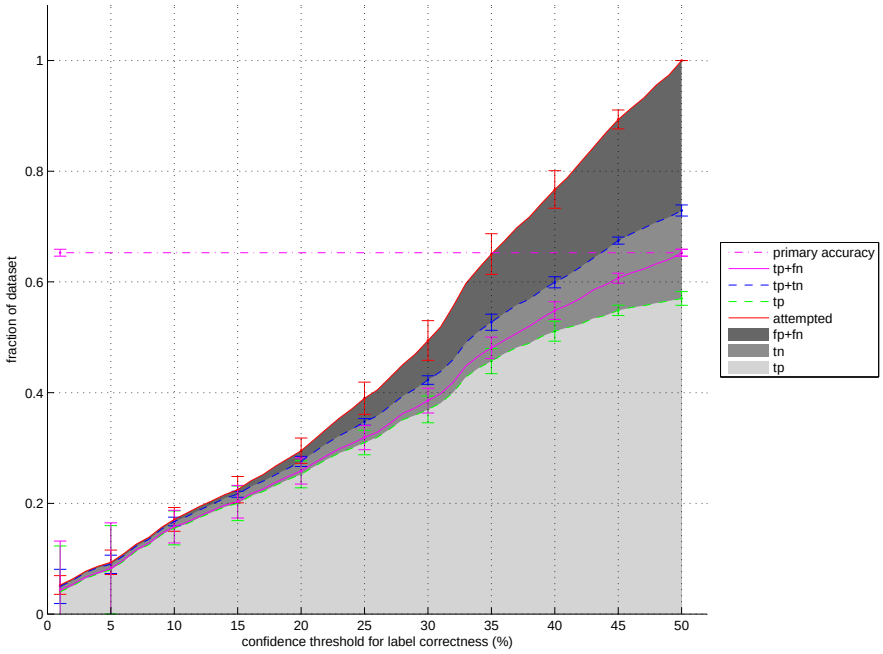


(A) Distance distributions and prediction confidence for codebook size 250

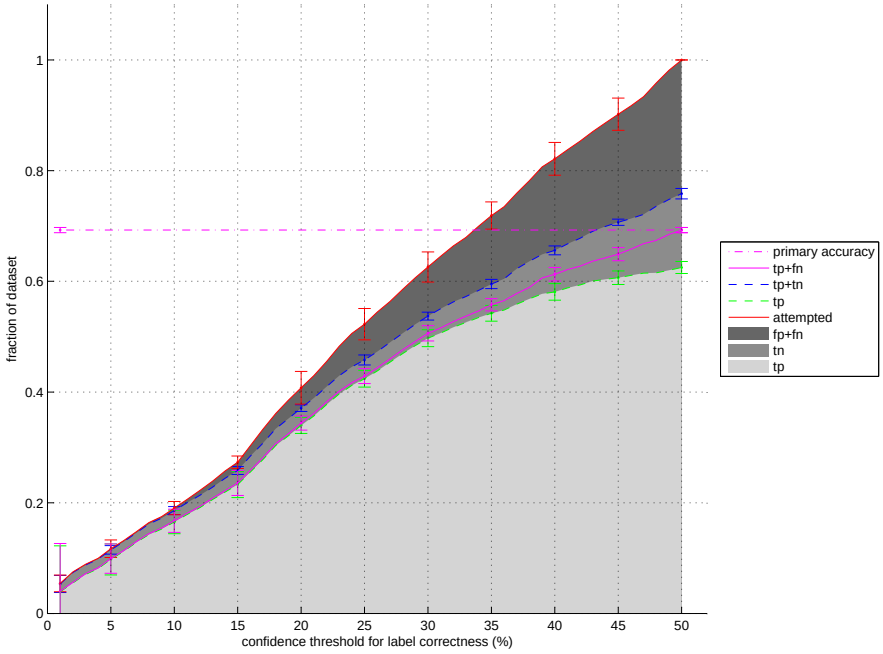


(B) Distance distributions and prediction confidence for codebook size 500

FIGURE D.12: Correct/Incorrect nearest-neighbour distance distributions for each dataset with prediction confidence, codebook sizes 250-500, eight runs

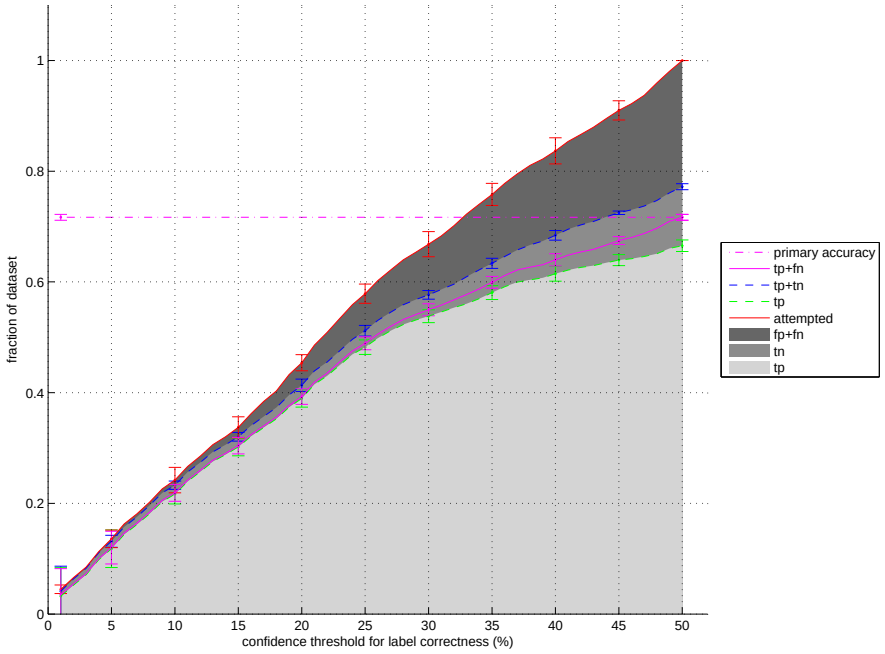


(A) Error/accuracy rates for codebook size 50

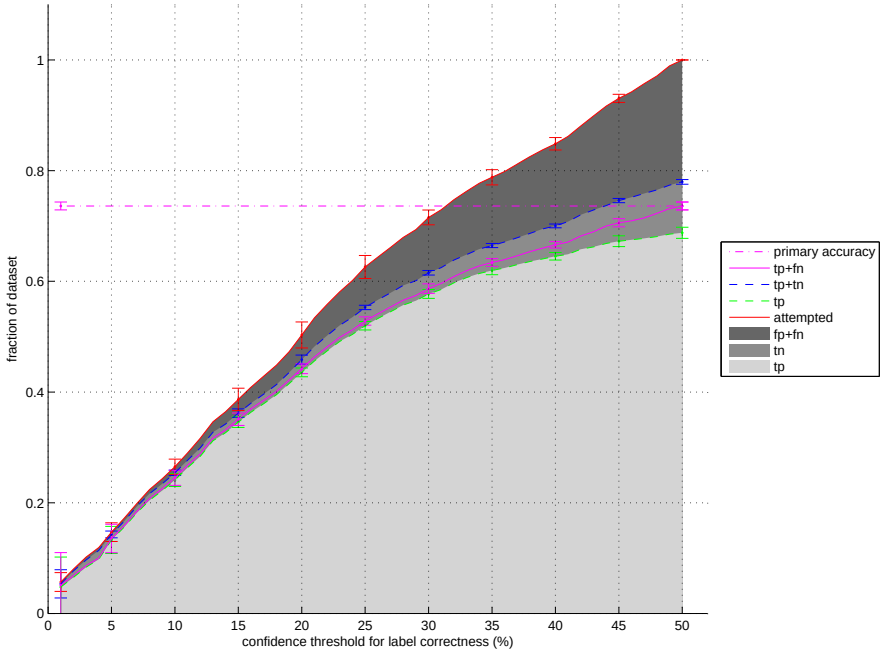


(B) Error/accuracy rates for codebook size 100

FIGURE D.13: Error rates for each confidence threshold as a proportion of the total dataset, codebook sizes 50-100, 8 runs mean and standard error

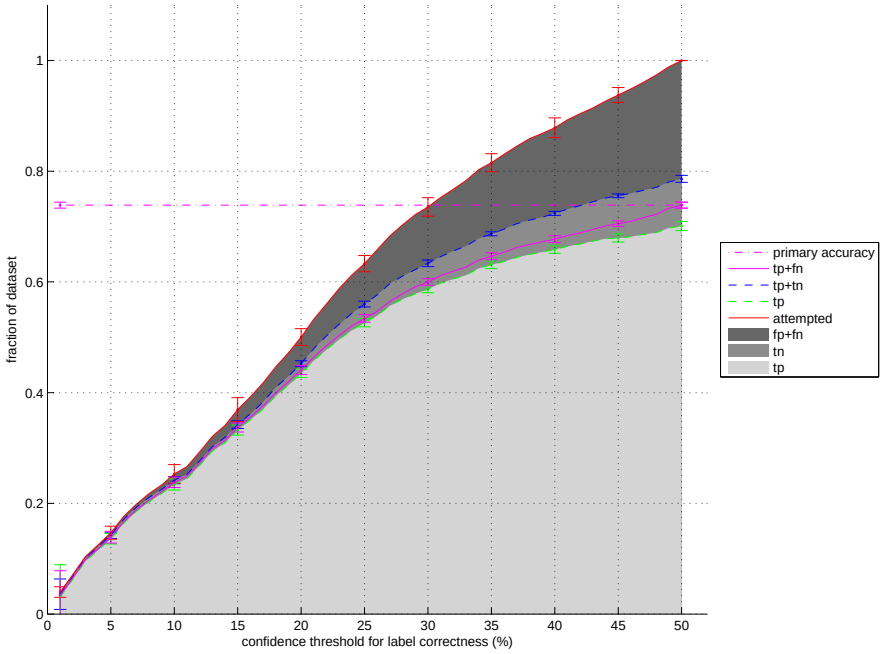


(A) Error/accuracy rates for codebook size 150

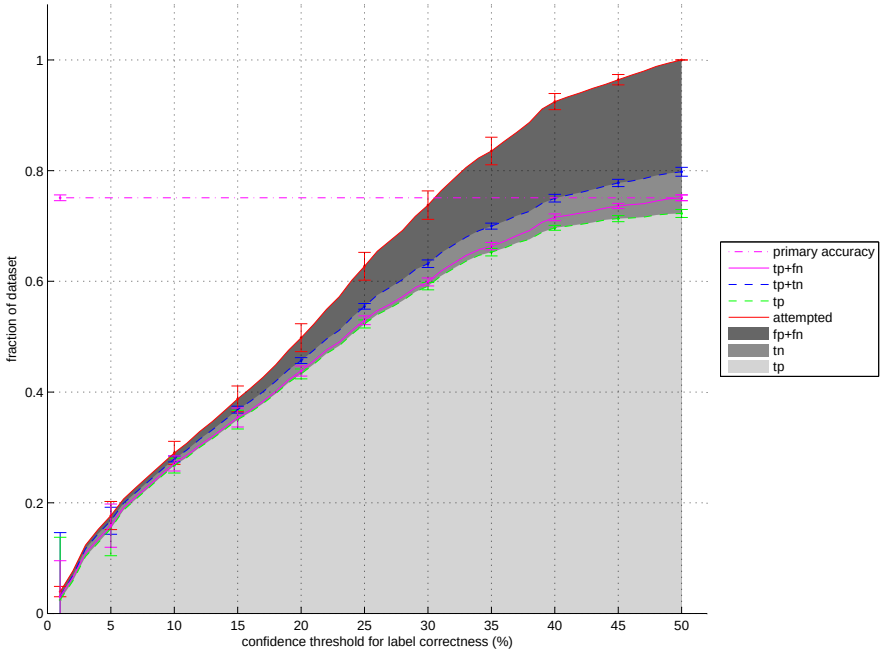


(B) Error/accuracy rates for codebook size 200

FIGURE D.14: Error rates for each confidence threshold as a proportion of the total dataset, codebook sizes 150-200, 8 runs mean and standard error

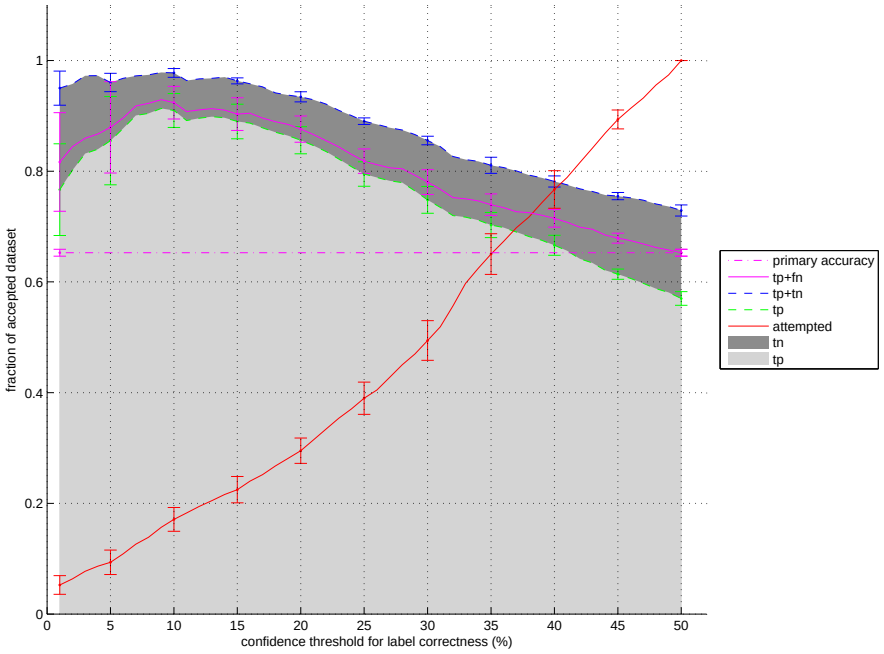


(A) Error/accuracy rates for codebook size 250

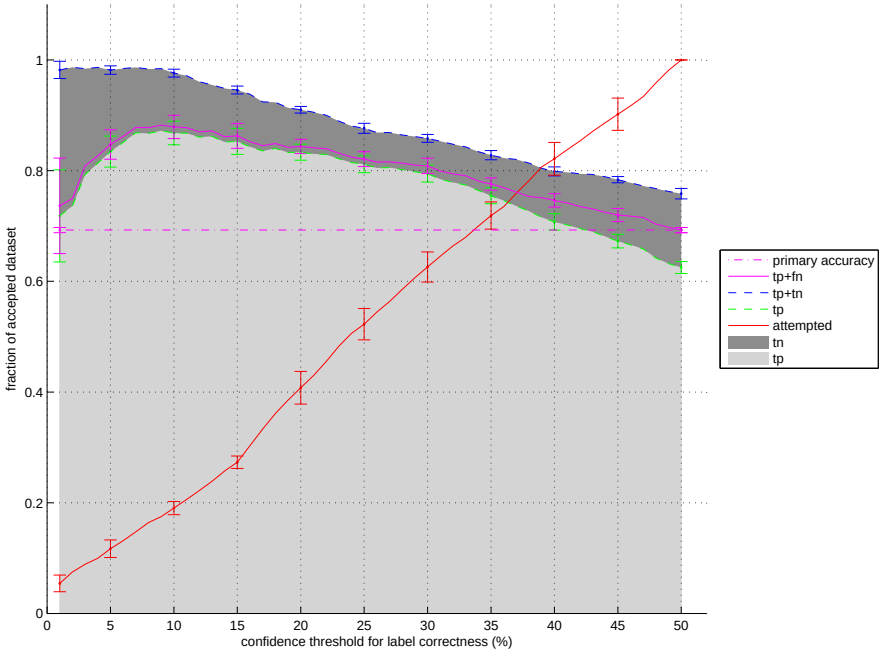


(B) Error/accuracy rates for codebook size 500

FIGURE D.15: Error rates for each confidence threshold as a proportion of the total dataset, codebook sizes 250-500, 8 runs mean and standard error

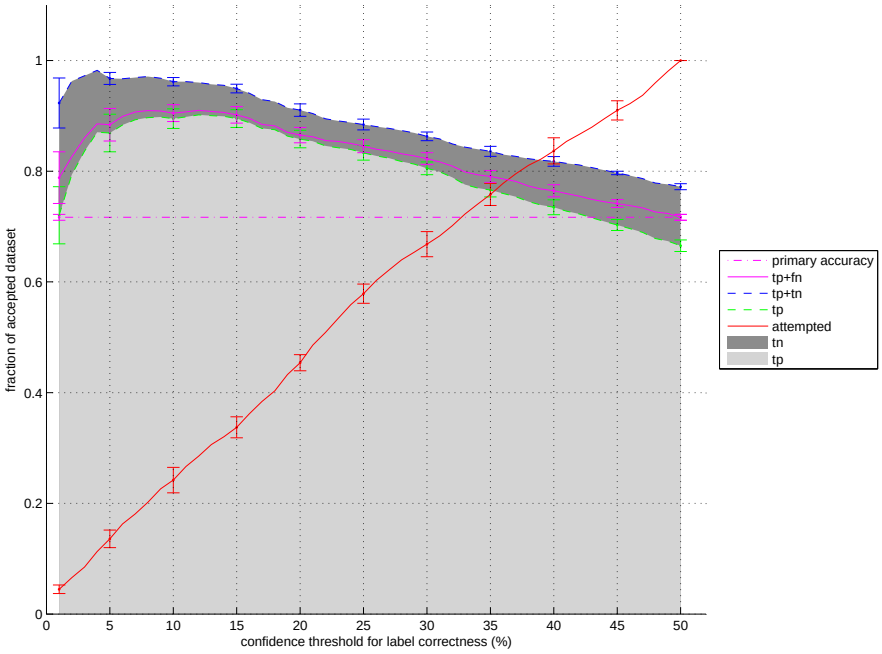


(A) Error/accuracy rates for codebook size 50

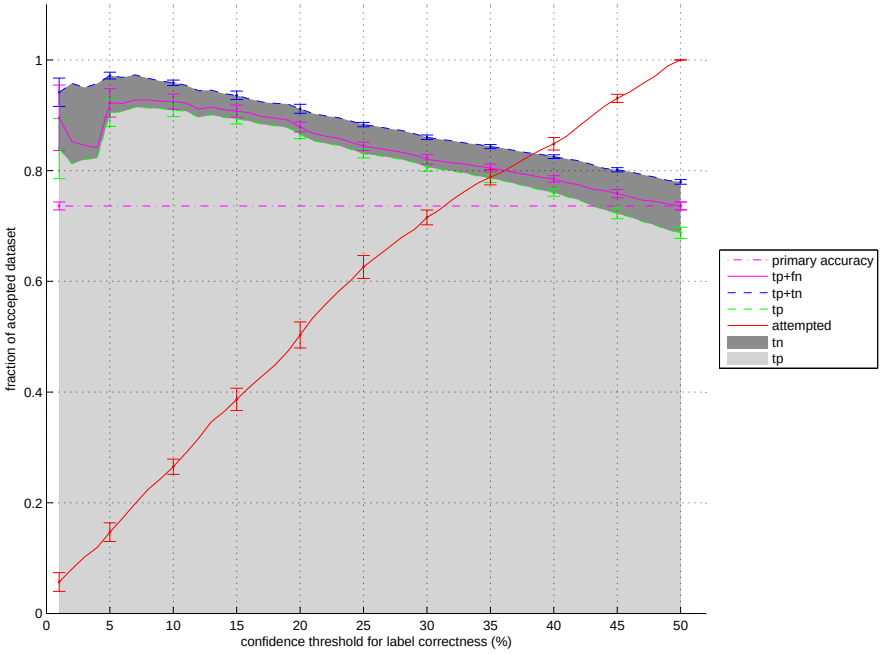


(B) Error/accuracy rates for codebook size 100

FIGURE D.16: Error rates for each confidence threshold as a proportion of the attempted dataset, codebook sizes 50-100, 8 runs mean and standard error

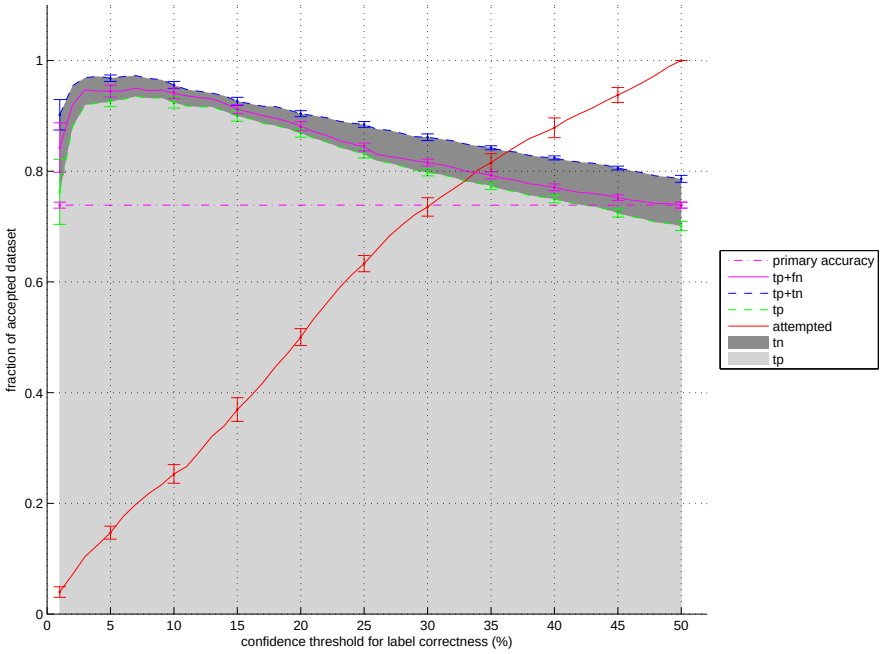


(A) Error/accuracy rates for codebook size 150

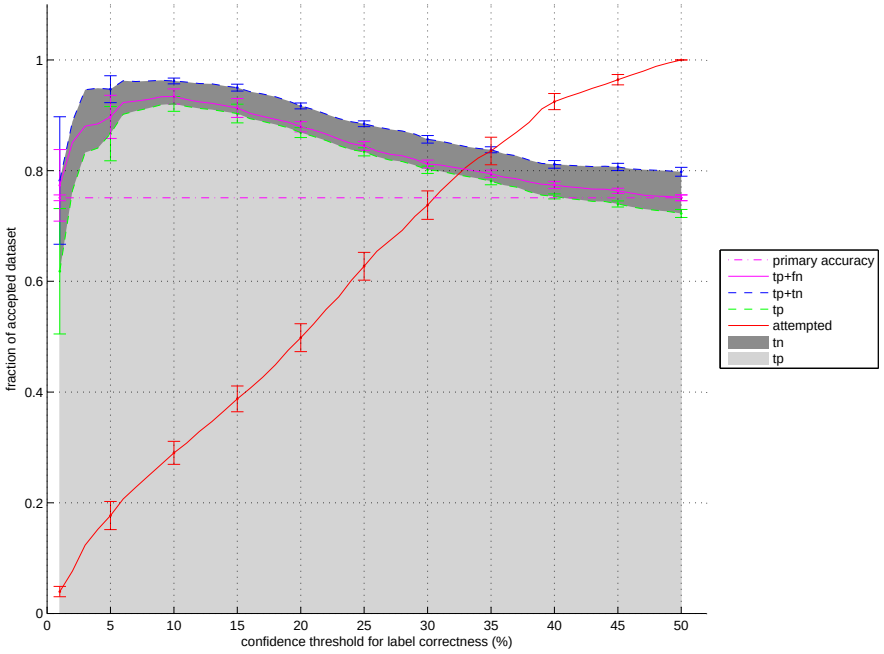


(B) Error/accuracy rates for codebook size 200

FIGURE D.17: Error rates for each confidence threshold as a proportion of the attempted dataset, codebook sizes 150-200, 8 runs mean and standard error



(A) Error/accuracy rates for codebook size 250



(B) Error/accuracy rates for codebook size 500

FIGURE D.18: Error rates for each confidence threshold as a proportion of the attempted dataset, codebook sizes 250-500, 8 runs mean and standard error

Bibliography

Mohamed N. Abdi, Maher Khemakhem, and Hanene Ben-Abdallah. A novel approach for off-line Arabic writer identification based on stroke feature combination. pages 597–600, 2009. doi: 10.1109/ISCIS.2009.5291888.

M. Amos. Identification of Medieval Scribal Handwriting. *Undergraduate thesis, unpublished*, 2004.

M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *Image Processing, IEEE Transactions on*, 1(2):205–220, 1992.

B. Arazi. Handwriting identification by means of run-length measurements. *IEEE Trans. Syst., Man and Cybernetics*, 7(12):878–881, 1977.

Itay Bar-Yosef, Isaac Beckman, Klara Kedem, and Itshak Dinstein. Binarization, character extraction, and writer identification of historical Hebrew calligraphy documents. *Int. J. Doc. Anal. Recognit.*, 9(2):89–99, 2007. doi: 10.1007/s10032-007-0041-5.

Michael F. Barnsley. *Fractals Everywhere*. Morgan Kaufmann, 2nd edition, April 2000. ISBN 0120790696.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957. URL `http://www.worldcat.org/isbn/0486428095`.

A. Bensefia, A. Nosary, T. Paquet, and L. Heutte. Writer identification by writer's invariants. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 274–279, 2002.

A. Bensefia, T. Paquet, and L. Heutte. Information retrieval based writer identification. In *7th International Conference on Document Analysis and Recognition*, pages 946–950, 2003.

Ameur Bensefia, Thierry Paquet, and Laurent Heutte. A writer identification and verification system. *Pattern Recognition Letters*, 26(13):2080–2092, 2005. doi: 10.1016/j.patrec.2005.03.024.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007. ISBN 0387310738.

Vivian Blankers, Ralph Niels, and Louis Vuurpijl. Writer identification by means of explainable features: shapes of loop and lead-in strokes. In *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2007)*, pages 17–24, 2007.

Ruud Bolle, Jonathan Connell, Sharanthchandra Pankanti, Nalini Ratha, and Andrew Senior. *Guide to Biometrics (Springer Professional Computing)*. Springer, 1st edition, November 2003. ISBN 0387400893.

Ronald Bracewell. *The Fourier Transform & Its Applications*. McGraw-Hill, 1st edition, 1965.

Anja Brakensiek, Andreas Kosmala, and Gerhard Rigoll. Comparing adaptation techniques for On-Line handwriting recognition. *Document Analysis and Recognition, International Conference on*, pages 486–490, 2001. doi: 10.1109/ICDAR.2001.953837.

K. Briechle and U. D. Hanebeck. Template Matching Using Fast Normalized Cross Correlation. In *Proceedings of SPIE: Optical Pattern Recognition XII*, volume 4387, pages 95–102, 2001. doi: 10.1117/12.421129.

Axel Brink, Lambert Schomaker, and Marius Bulacu. Towards Explainable Writer Verification and Identification Using Vantage Writers. In *Document Analysis and Recognition, 2007. ICDAR 2007 Vol. 2. Ninth International Conference on*, volume 2, pages 824–828, 2007.

Axel Brink, Marius Bulacu, and Lambert Schomaker. How much handwritten text is needed for text-independent writer verification and identification. In *Proc. of 19th Int. Conf. on Pattern Recognition (ICPR 2008)*, 2008.

Michael Bryant, Tobias Blanke, Mark Hedges, and Richard Palmer. Open source historical OCR: The OCRopodium project. In Mounia Lalmas, Joemon Jose, Andreas Rauber, Fabrizio Sebastiani, and Ingo Frommholz, editors, *Research and Advanced Technology for Digital Libraries*, volume 6273 of *Lecture Notes in Computer Science*, chapter 72, pages 522–525. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010. doi: 10.1007/978-3-642-15464-5_72.

M. Bulacu and L. Schomaker. Writer style from oriented edge fragments. In *Proc. of the 10th Int. Conference on Computer Analysis of Images and Patterns*, pages 460–469, 2003.

M. Bulacu and L. Schomaker. Combining Multiple Features for Text-Independent Writer Identification and Verification. pages 281–286, 2006.

M. Bulacu, L. Schomaker, and L. Vuurpijl. Writer identification using edge-based directional features. In *7th International Conference on Document Analysis and Recognition*, 2003.

M. Bulacu, L. Schomaker, and A. Brink. Text-Independent Writer Identification and Verification on Offline Arabic Handwriting. In *9th International Confer-*

ence on Document Analysis and Recognition, volume 2, pages 769–773, 2007a. doi: 10.1109/ICDAR.2007.4377019.

Marius Bulacu. *Statistical pattern recognition for automatic writer identification and verification*. PhD thesis, 2007.

Marius Bulacu and Lambert Schomaker. A Comparison of Clustering Methods for Writer Identification and Verification. In *8th International Conference on Document Analysis and Recognition*, pages 1275–1279, Washington, DC, USA, 2005a. IEEE Computer Society.

Marius Bulacu and Lambert Schomaker. Grawis: Groningen automatic writer identification system. In *Proc. of 17th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2005)*, pages 413–414, 2005b.

Marius Bulacu and Lambert Schomaker. Automatic handwriting identification on medieval documents. In *Proc. of 14th Int. Conf. on Image Analysis and Processing (ICIAP 2007)*, pages 279–284, 2007a.

Marius Bulacu and Lambert Schomaker. Text-Independent Writer Identification and Verification Using Textural and Allographic Features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(4):701–717, 2007b.

Marius Bulacu, Rutger van Koert, Lambert Schomaker, and Tijn van der Zant. Layout Analysis of Handwritten Historical Documents for Searching the Archive of the Cabinet of the Dutch Queen. In *9th International Conference on Document Analysis and Recognition*, volume 1, pages 357–361, 2007b.

R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):690–706, 1996. doi: 10.1109/34.506792.

Siew K. Chan, Yong H. Tay, and Christian C. Viard-Gaudin. Online text independent writer identification using character prototypes distribution. In

2007 6th International Conference on Information, Communications & Signal Processing, pages 1–5. IEEE, 2007. ISBN 978-1-4244-0982-2. doi: 10.1109/ICICS.2007.4449745.

Joulia Chapran. Biometric Writer Identification: Feature Analysis And Classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(4):483–503, 2006.

Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, and Ching Suen. *Character Recognition Systems: A Guide for Students and Practitioners*. Wiley-Interscience, October 2007. ISBN 0471415707.

Arianna Ciula. Digital palaeography: using the digital representation of medieval script to support palaeographic analysis. *Digital Medievalist*, April 2005. URL `http://www.digitalmedievalist.org/journal/1.1/ciula/`.

John Daugman. Biometric decision landscapes. Technical report, 2000.

John Daugman. Recognising persons by their iris patterns. In *Advances in Biometric Person Authentication*, pages 5–25. 2005.

Tom Davis. The Practice of Handwriting Identification. *Library*, 8(3):251–276, 2007. doi: 10.1093/library/8.3.251.

C. Dibben. Identification of Medieval Scribal Handwriting. *Undergraduate thesis, unpublished*, 2004.

Due, Anil K. Jain, and Torfinn Taxt. Feature extraction methods for character recognition-A survey. *Pattern Recognition*, 29(4):641–662, 1996. doi: 10.1016/0031-3203(95)00118-2.

Ted Dunstone and Neil Yager. *Biometric System and Data Analysis Design, Evaluation, and Data Mining*. Springer, 2009.

R. Fallon. Identification of Medieval Scribal Handwriting (Implementation). *Undergraduate thesis, unpublished*, 2004.

Andreas Fischer, Markus Wuthrich, Marcus Liwicki, Volkmar Frinken, Horst Bunke, Gabriel Viehhauser, and Michael Stolz. Automatic Transcription of Handwritten Medieval Documents. pages 137–142, 2009. doi: 10.1109/VSMM.2009.26.

Andreas Fischer, Kaspar Riesen, and Horst Bunke. Graph Similarity Features for HMM-Based Handwriting Recognition in Historical Documents. *Frontiers in Handwriting Recognition, International Conference on*, 0:253–258, 2010. doi: 10.1109/ICFHR.2010.47.

Alicia Fornéz, Josep Lladós, Gemma Sánchez, and Horst Bunke. On the Use of Textural Features for Writer Identification in Old Handwritten Music Scores. In *10th International Conference on Document Analysis and Recognition*, pages 996–1000, 2009. doi: 10.1109/ICDAR.2009.100.

Katrin Franke, Lambert Schomaker, Christian Veenhuis, Louis Vuurpijl, Merijn van Erp, and Isabelle Guyon. WANDA: A common ground for forensic handwriting examination and writer identification. Technical report, 2003.

Utpal Garain and Thierry Paquet. Off-Line Multi-Script Writer Identification Using AR Coefficients. pages 991–995, 2009. doi: 10.1109/ICDAR.2009.222.

Sonia Garcia-Salicetti, Charles Beumier, Gérard Chollet, Bernadette Dorizzi, Jean Jardins, Jan Lunter, Yang Ni, and Dijana Petrovska-Delacrétaz. BIOMET: A multimodal person authentication database including face, voice, fingerprint, hand and signature modalities. In Josef Kittler and Mark Nixon, editors, *Audio- and Video-Based Biometric Person Authentication*, volume 2688 of *Lecture Notes in Computer Science*, chapter 98, page 1056. Springer Berlin / Heidelberg, June 2003.

S. Gazzah and N. Ben Amara. Arabic Handwriting Texture Analysis for Writer Identification Using the DWT-Lifting Scheme. In *9th International Conference on Document Analysis and Recognition*, volume 2, pages 1133–1137, 2007.

W. M. Gentleman and G. Sande. Fast Fourier Transforms: for fun and profit. AFIPS '66 (Fall), pages 563–578. ACM, 1966. doi: 10.1145/1464291.1464352.

Golnaz Ghiasi and Reza Safabakhsh. An Efficient Method for Offline Text Independent Writer Identification. pages 1245–1248, 2010. doi: 10.1109/ICPR.2010.310.

Alexandra Gillespie and Daniel Wakelin, editors. *The production of books in England 1350-1500*. Cambridge University Press, 2011.

David C. Greetham. *Textual scholarship: an introduction*. Garland Reference, corrected edition, 1994.

Maya R. Gupta, Nathaniel P. Jacobson, and Eric K. Garcia. OCR binarization and image pre-processing for searching historical documents. *Pattern Recognition*, 40(2):389–397, 2007. doi: 10.1016/j.patcog.2006.04.043.

I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. UNIPEN project of on-line data exchange and recognizer benchmarks. In *Proceedings of the 12th IAPR International Conference on Computer Vision & Image Processing*, volume 2, pages 29–33, 1994.

Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3 (6):610–621, 1973. doi: 10.1109/TSMC.1973.4309314.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

Zhenyu He, Bin Fang, Jianwei Du, Yuan Y. Tang, and Xinge You. A novel method for offline handwriting-based writer identification. In *8th International Conference on Document Analysis and Recognition*, volume 1, pages 242–246, 2005.

Behzad Helli and Mohsen E. Moghaddam. A text-independent Persian writer identification based on feature relation graph (FRG). *Pattern Recognition*, 2009. doi: 10.1016/j.patcog.2009.11.026.

Caroline Hertel and Horst Bunke. *A Set of Novel Features for Writer Identification*, pages 679–687. SpringerLink, Lecture Notes in Computer Science, 2003.

G. Hughes. On the mean accuracy of statistical pattern recognizers. 14(1):55–63, January 1968. ISSN 0018-9448. doi: 10.1109/TIT.1968.1054102.

John Hutchinson. Fractals and Self-Similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

Aapo Hyvärinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural Comput.*, 9:1483–1492, October 1997. ISSN 0899-7667.

Aapo Hyvarinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. Wiley-Interscience, 1 edition, 2001. ISBN 047140540X.

A. Imdad, S. Bres, V. Eglin, C. Rivero-Moreno, and H. Emptoz. Writer Identification Using Steered Hermite Features and SVM. In *Document Analysis and Recognition, 2007. ICDAR 2007 Vol. 2. Ninth International Conference on*, volume 2, pages 839–843, 2007.

Donato Impedovo and Giuseppe Pirlo. Automatic Signature Verification: The State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5):609–635, 2008. doi: 10.1109/TSMCC.2008.923866.

A. Jain. On-line signature verification. *Pattern Recognition*, 35(12):2963–2972, 2002. doi: 10.1016/S0031-3203(01)00240-0.

A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20, 2004.

Anil K. Jain, Ruud M. Bolle, and Sharath Pankanti. *Biometrics: Personal Identification in Networked Society*. Springer, 2005.

Anil K. Jain, Patrick Flynn, and Arun A. Ross, editors. *Handbook of Biometrics*. Springer, 2008.

Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, Inc., 1990. doi: 10.1002/9780470316801.ch1.

Ergina Kavallieratou. A Binarization Algorithm specialized on Document Images and Photos. In *8th International Conference on Document Analysis and Recognition*, pages 463–467, Washington, DC, USA, 2005. IEEE Computer Society. doi: 10.1109/ICDAR.2005.1.

Eiji Kawaguchi. BPCS-Steganography Principle and Applications. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 289–299. 2005. doi: 10.1007/11554028_41.

Nevena Lazic and Parham Aarabi. Importance of Feature Locations in Bag-of-Words Image Classification. pages I–641–I–644, 2007. doi: 10.1109/ICASSP.2007.365989.

F. Leclerc and R. Plamondon. Automatic signature verification: the state of the art-1989-1993. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 8(3):643–660, 1994.

C. K. Lee and C. G. Leedham. A new hybrid approach to handwritten address verification. *International Journal of Computer Vision*, 57:107–120, May 2004.

Graham Leedham, Saket Varma, Anish Patankar, and Venu Govindarayu. Separating Text and Background in Degraded Document Images: A Comparison of Global Threshholding Techniques for Multi-Stage Threshholding. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, Washington, DC, USA, 2002. IEEE Computer Society.

Bangy Li and Tieniu Tan. Online Text-independent Writer Identification Based on Temporal Sequence and Shape Codes. pages 931–935, 2009. doi: 10.1109/ICDAR.2009.26.

Bangyu Li, Zhenan Sun, and Tieniu Tan. Online Text-Independent Writer Identification Based on Stroke's Probability Distribution Function. In *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*, chapter 22, pages 201–210. Springer Berlin / Heidelberg, 2007.

Fei-Fei Li and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 524–531, 2005.

Laurence Likforman-Sulem, Abderrazak Zahour, and Bruno Taconet. Text line segmentation of historical documents: a survey. *International Journal on Document Analysis and Recognition*, 9(2):123–138, 2007.

C. Liu. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, October 2003. ISSN 00313203. doi: 10.1016/S0031-3203(03)00085-2.

David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 1st edition, October 2003. ISBN 0521642981.

URL http://www.inference.phy.cam.ac.uk/mackay/itila/book.html.

H. Maclean. Identification of Medieval Scribal Handwriting. *Undergraduate thesis, unpublished*, 2004.

R. Manmatha, Chengfeng Han, E. M. Riseman, and W. B. Croft. Indexing handwriting using word matching. In *Proceedings of the first ACM international conference on Digital libraries*, pages 151–159, 1996. doi: 10.1145/226931.226960.

V. Märgner and H. El Abed. Arabic Handwriting Recognition Competition. pages 1274–1278, 2007. doi: 10.1109/ICDAR.2007.4377120.

V. Märgner, M. Pechwitz, and H. El Abed. Arabic Handwriting Recognition Competition. *Document Analysis and Recognition, International Conference on*, 0: 70–74, 2005. doi: 10.1109/ICDAR.2005.52.

Volker Märgner and H. El Abed. ICDAR 2009 Arabic Handwriting Recognition Competition. pages 1383–1387, 2009. doi: 10.1109/ICDAR.2009.256.

Simone Marinai, Beatrice Miotti, and Giovanni Soda. Bag of Characters and SOM Clustering for Script Recognition and Writer Identification. pages 2182–2185, 2010. doi: 10.1109/ICPR.2010.534.

U. V. Marti and H. Bunke. A full English sentence database for off-line handwriting recognition. In *Proceedings of the 5th International Conference on Document Analysis and Recognition*, pages 705–708, 1999. doi: 10.1109/ICDAR.1999.791885.

U. V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems. *International Journal of Pattern Recognition and Artificial Intelligence*, pages 65–90, 2002. doi: 10.1142/S0218001401000848.

U. V. Marti, R. Messerli, and H. Bunke. Writer identification using text line based features. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 101–105, 2001.

Linne R. Mooney. Chaucer's Scribe. *Speculum*, 81(01):97–138, 2006. doi: 10.1017/S0038713400019394.

Shunji Mori, Hirobumi Nishida, and Hiromitsu Yamada. *Optical Character Recognition (Wiley Series in Microwave and Optical Engineering)*. Wiley-Interscience, April 1999. ISBN 0471308196.

Anoop Namboodiri and Sachin Gupta. Text Independent Writer Identification from Online Handwriting. In *Tenth International Workshop on Frontiers in Handwriting Recognition, La Baule, France*, 2006.

F. Nejad and M. Rahmati. A New Method for Writer Identification and Verification Based on Farsi/Arabic Handwritten Texts. In *9th International Conference on Document Analysis and Recognition*, volume 2, pages 829–833, 2007.

Wayne Niblack. *An Introduction to Digital Image Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

Ralph Niels, Louis Vuurpijl, and Lambert Schomaker. Introducing TRIGRAPH - Trimodal writer identification.

A. Nosary, L. Heutte, T. Paquet, and Y. Lecourtier. Defining writer's invariants to adapt the recognition task. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 765–768, 1999.

J. Ortega-Garcia, J. Fierrez-Aguilar, D. Simon, J. Gonzalez, M. Faundez-Zanuy, V. Espinosa, A. Satue, I. Hernaez, J. J. Igarza, C. Vivaracho, D. Escudero, and Q. I. Moro. MCYT baseline corpus: a bimodal biometric database. *Vision, Image and Signal Processing*, 150(6):395–401, December 2003.

N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

Rudolf Pareti and Nicole Vincent. Global Method Based on Pattern Occurrences for Writer Identification. In *Tenth International Workshop on Frontiers in Handwriting Recognition (2006)*, 2006.

Mario Pechwitz, Samia S. Maddouri, Volker Märgner, Noureddine Ellouze, and Hamid Amiri. IFN/ENIT - database of handwritten Arabic words. In *In Proc. of CIFED 2002*, pages 129–136, 2002.

Vladimir Pervouchine and Graham Leedham. Extraction and Analysis of Document Examiner Features from Vector Skeletons of Grapheme 'th'. *Document Analysis Systems VII*, pages 196–207, 2006.

Vladimir Pervouchine and Graham Leedham. Extraction and analysis of forensic document examiner features used for writer identification. *Pattern Recognition*, 40(3):1004–1013, 2007.

Rejean Plamondon and Guy Lorette. Automatic signature verification and writer identification – the state of the art. *Pattern Recognition*, 22(2):107–131, 1989. doi: 10.1016/0031-3203(89)90059-9.

Réjean Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, November 1994.

S. Sadeghi ram and M. E. Moghaddam. A Persian Writer Identification Method Based on Gradient Features and Neural Networks. pages 1–4, 2009. doi: 10.1109/CISP.2009.5301092.

H. E. S. Said, K. D. Baker, and T. N. Tan. Personal identification based on handwriting. In *Fourteenth International Conference on Pattern Recognition*, volume 2, pages 1761–1764, 1998. doi: 10.1109/ICPR.1998.712068.

A. Schlapbach and H. Bunke. Off-line handwriting identification using HMM based recognizers. volume 2, 2004a.

A. Schlapbach and H. Bunke. Using HMM based recognizers for writer identification and verification. pages 167–172, 2004b.

A. Schlapbach and H. Bunke. Off-lineWriter Identification Using Gaussian Mixture Models. In *18th International Conference on Pattern Recognition*, volume 3, pages 992–995, 2006a.

A. Schlapbach and H. Bunke. Fusing Asynchronous Feature Streams for On-line Writer Identification. In *9th International Conference on Document Analysis and Recognition*, volume 1, pages 103–107, 2007a.

Andreas Schlapbach and Horst Bunke. Writer Identification Using an HMM-Based Handwriting Recognition System: To Normalize the Input or Not? In *Proc. 12th Conf. of the Int. Graphonomics Society*, pages 138–142, 2005.

Andreas Schlapbach and Horst Bunke. Off-Line Writer Verification: A Comparison of a Hidden Markov Model (HMM) and a Gaussian Mixture Model (GMM) Based System. In *Tenth International Workshop on Frontiers in Handwriting Recognition (2006)*, 2006b.

Andreas Schlapbach and Horst Bunke. A writer identification and verification system using HMM based recognizers. *Pattern Analysis and Applications (PAA)*, 10(1):33–43, 2007b. doi: 10.1007/s10044-006-0047-5.

Andreas Schlapbach, Vivian Kilchherr, and Horst Bunke. Improving Writer Identification by Means of Feature Selection and Extraction. In *Eighth Interna-*

tional Conference on Document Analysis and Recognition (ICDAR'05), pages 131–135, 2005.

Lambert Schomaker. Advances in Writer identification and verification. In *ICDAR07*, 2007.

Lambert Schomaker and Marius Bulacu. Automatic Writer Identification Using Connected-Component Contours and Edge-Based Features of Upper-Case Western Script. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):787–798, 2004.

Lambert Schomaker, Marius Bulacu, and Katrin Franke. Automatic Writer Identification Using Fragmented Connected-Component Contours. In *Proc. of 9th International Workshop on Frontiers in Handwriting Recognition (IWHFR04)*, pages 185–190, 2004.

Lambert Schomaker, Katrin Franke, and Marius Bulacu. Using codebooks of fragmented connected-component contours in forensic and historic writer identification. *Pattern Recognition Letters*, 28(6):719–727, 2007.

A. Seropian, M. Grimaldi, and N. Vincent. Writer identification based on the fractal construction of a reference base. In *7th International Conference on Document Analysis and Recognition*, pages 1163–1167, 2003.

F. Shahabi and M. Rahmati. Comparison of Gabor-Based Features for Writer Identification of Farsi/Arabic Handwriting. In *Proceedings of 10th IWFHR*, pages 545–550, 2006.

F. Shahabi and M. Rahmati. A New Method for Writer Identification of Handwritten Farsi Documents. In *10th International Conference on Document Analysis and Recognition*, pages 426–430, 2009. doi: 10.1109/ICDAR.2009.290.

I. Siddiqi and N. Vincent. Writer Identification in Handwritten Documents. In *9th International Conference on Document Analysis and Recognition*, volume 1, pages 108–112, 2007.

Sreeraj and Sumam M. Idicula. A survey on writer identification schemes. *International Journal of Computer Applications*, 26(2):23–33, July 2011. doi: 10.5120/3075-4205.

S. N. Srihari, S. H. Cha, H. Arora, and S. Lee. Individuality of handwriting. *Journal of Forensic Sciences*, pages 1–17, 2002.

S. N. Srihari, C. I. Tomai, Bin Zhang, and Sangjik Lee. Individuality of numerals. In *7th International Conference on Document Analysis and Recognition*, pages 1096–1100, 2003.

Sargur N. Srihari and Graham Leedham. A survey of computer methods in forensic document examination. In *Proceedings of 11th Conference International on Graphonomics Society (IGS2003)*, pages 278–281, 2003.

Geetha Srikantan, Stephen W. Lam, and Sargur N. Srihari. Gradient-based contour encoding for character recognition. *Pattern Recognition*, 29(7):1147–1160, 1996. doi: 10.1016/0031-3203(95)00146-8.

Peter Stokes. *Computer-Aided Palaeography, Present and Future*. Institut für Dokumentologie und Editorik, 2009.

Peter A. Stokes. Palaeography and Image-Processing: Some solutions and problems. *Digital Medievalist*, December 2007. URL `http://www.digitalmedievalist.org/journal/3/stokes/`.

C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE*, 80(7):1162–1180, July 1992. ISSN 0018-9219. doi: 10.1109/5.156477.

P. J. Sutanto, G. Leedham, and V. Pervouchine. Study of the consistency of some discriminatory features used by document examiners in the analysis of handwritten letter 'a'. In *7th International Conference on Document Analysis and Recognition*, pages 1091–1095, 2003.

Guo X. Tan, Christian V. Gaudin, and Alex C. Kot. Impact of Alphabet Knowledge on Online Writer Identification. *Document Analysis and Recognition, International Conference on*, 0:56–60, 2009. doi: 10.1109/ICDAR.2009.165.

Marino Tapiador and Juan A. Sigüenza. Writer Identification Method Based on Forensic Knowledge. In David Zhang and Anil K. Jain, editors, *Biometric Authentication*, volume 3072 of *Lecture Notes in Computer Science*, chapter 76, pages 555–561. Springer Berlin / Heidelberg, 2004.

C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(8):787–808, August 1990. ISSN 0162-8828. doi: 10.1109/34.57669.

Natasa Terzija and Walter Geisselhardt. Digital image watermarking using complex wavelet transform. In *Proceedings of the 2004 Workshop on Multimedia and Security*, pages 193–198, New York, NY, USA, 2004. ACM. doi: 10.1145/1022431.1022465.

Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 3rd edition, March 2006. ISBN 0123695317.

Kurban Ubul, Dilmurat Tursun, Askar Hamdulla, and Alim Aysa. A Feature Selection and Extraction Method for Uyghur Handwriting-Based Writer identification. pages 345–348, 2009. doi: 10.1109/CINC.2009.198.

L. J. P. van der Maaten. Improving Automatic Writer Identification. Master's thesis, Maastricht University, 2005.

L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. Technical report, 2009.

Laurens van der Maaten and Eric Postma. Improving Automatic Writer Identification. In *Proceedings of the Seventeenth Belgian/Dutch Conference on AI (BNAIC05)*, pages 260–266, 2005.

M. van Erp, L. Schomaker, and M. Bulacu. Sparse-Parametric Writer Identification Using Heterogeneous Feature Groups. In *Proc. of Int. Conf. on Image Processing (ICIP 2003)*, volume 1, pages 545–548, 2003.

Judie Walton. Handwriting changes due to aging and Parkinson's syndrome. *Forensic Science International*, 88(3):197–214, 1997. doi: 10.1016/S0379-0738(97)00105-9.

Xianliang Wang, Xiaoqing Ding, and Hailong Liu. Writer identification using directional element features and linear transform. In *7th International Conference on Document Analysis and Recognition*, pages 942–945, 2003.

M. Wirotius, A. Seropian, and N. Vincent. Writer identification from gray level distribution. In *7th International Conference on Document Analysis and Recognition*, pages 1168–1172, 2003.

Jeffrey Woodard, Mark Lancaster, Amlan Kundu, Dan Ruiz, and John Ryan. Writer recognition of Arabic text by generative local features. pages 1–7, 2010. doi: 10.1109/BTAS.2010.5634495.

Dit-yan Yeung, Hong Chang, Yimin Xiong, Susan George, Ramanujan Kashi, Takashi Matsumoto, and Gerhard Rigoll. SVC2004: First international signature verification competition. In *In Proceedings of the International Conference on Biometric Authentication (ICBA), Hong Kong*, volume 3072, pages 16–22, 2004.

Bin Zhang and S. N. Srihari. Analysis of handwriting individuality using word features. In *7th International Conference on Document Analysis and Recognition*, pages 1142–1146, 2003.

Bin Zhang, S. N. Srihari, and Sangjik Lee. Individuality of handwritten characters. In *7th International Conference on Document Analysis and Recognition*, pages 1086–1090, 2003.

Yong Zhu, Tieniu Tan, and Yunhong Wang. Biometric personal identification based on handwriting. In *15th International Conference on Pattern Recognition*, pages 797–800, 2000.

E. N. Zois and V. Anastassopoulos. Methods for writer identification. In *Proceedings of the 3rd IEEE International Conference on Electronics, Circuits, and Systems*, volume 2, pages 740–743, 1996.

Long Zuo, Yunhong Wang, and Tieniu Tan. Personal Handwriting Identification Based on PCA. In *Proceedings of SPIE Second International Conference on Image and Graphics*, pages 766–771, 2002.